MASTER EN SCIENCES INFORMATIQUES

Composition automatique de frottis numériques: application aux giga-images

Decock, Guillaume

Award date:
2004

Facultés Universitaires Notre-Dame de la Paix, Namur
Institut d'Informatique
Année académique 2003-2004

Composition automatique de frottis numériques : application aux giga-images

par Guillaume Decock

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique

Résumé :

Ce mémoire décrit une tentative de solution permettant la réalisation automatique de « frottis numériques ». Cette solution est présentée au travers d'une application basée d'une part, sur les phases principales que sont l'acquisition, l'assemblage et le stockage d'un ensemble d'images et d'autre part sur les concepts de « méga-images », « giga-images » et de coregistration, car dès qu'il s'agit de télémicroscopie, le volume de données à manipuler prend une ampleur considérable.

Ce document met aussi en évidence l'importance que représente la méthode de compression Jpeg2000 dans le domaine de la télémicroscopie. Et notamment l'importance des fonctionnalités destinées à la manipulation des grandes images.

Mots clés :

Télémicroscopie, frottis numériques, Jpeg2000, méga-image, giga-image, coregistration

Abstract :

This memoir describes an attempt of solution allowing the realisation of a "numerical smear". This solution is introduced through an application based on the one hand, on the main phases that are the acquisition, assembly and storage of a set of images. On the other hand on the concepts of "mega-image", "giga-image" and coregistration because as soon as telemicroscopy is concerned, the volume of data to handle is considerable.

This document shows the importance of the Jpeg2000 method compression in domain like telemicroscopy. And particularly the importance of the functionalities intended to the handling of big images.

Key words:

Telemicroscopy, numerical smear, Jpeg2000, mega-image, giga-image, coregistration

Remerciements :

Je tiens tout d'abord à remercier Monsieur Jean-Paul Leclerq, qui a accepté d'être le promoteur de ce mémoire et qui m'a suivi et encouragé lors de la rédaction de celui-ci.

Je remercie également Monsieur Hubert Meurisse qui en tant que mon maître de stage et co-promoteur, m'a aidé tout au long de mon stage. Je le remercie aussi pour sa participation et sa collaboration lors de la réalisation de cet ouvrage.

J'aimerais exprimer ma reconnaissance à Monsieur Louis Zuyderhoff pour sa patience face à mes nombreuses questions, ainsi que pour ses judicieux conseils lors de l'élaboration de mon mémoire.

Je remercie aussi le personnel du laboratoire d'hématologie et de cytologie de la Clinique Universitaire de Mont Godinne et particulièrement Monsieur Yvan Cornet pour sa disponibilité ainsi que pour son expérience dans le domaine de la cytologie hématologique.

Je me dois aussi de remercier mon collègue de la « petite pièce » David Devrees, qui m'a aidé à maintes reprises durant mes phases de codage. C'est aussi grâce à lui que j'ai passé de fabuleuses pauses de midi avec les canards de la petite mare.

Je voudrais aussi dire merci à tous mes amis avec lesquels j'ai passé cinq années fantastiques et ce aussi bien dans les auditoires que dans le bunker. Je fais un clin d'œil particulier à mon pote f®ed pour tous les bons moments qu'on a vécu.

Enfin, je remercie mes proches et spécialement mes parents qui m'ont soutenu et encouragé durant ces cinq années.

Introduction

Les évolutions sans cesse croissantes des technologies de l'information et des télécommunications ont contribué à l'élaboration de nombreuses applications fournissant des services de santé à distance regroupés sous le terme **télémédecine**. Ces évolutions ont notamment permis et facilité la consultation et l'échange d'informations entre experts. C'est dans ce cadre que la **télémicroscopie** a vu le jour.

La télémicroscopie comme son nom l'indique provient de l'association entre la microscopie et les technologies de l'information et des télécommunications. Les sciences telles que la cytologie et l'histologie trouvent dans la télémicroscopie des avantages formidables par rapport à la microscopie traditionnelle.

Un de ces avantages s'apprécie par exemple lors de l'analyse de cellules ou de tissus anormaux. En effet, la mise sous format électronique des lames par l'utilisation de la télémicroscopie présente l'avantage d'augmenter la fréquence, la rapidité et la qualité des discussions entre biologistes.

La numérisation d'une lame présente aussi d'autres avantages :

Il faut savoir qu'avant de pouvoir être analysée au microscope, une lame doit subir une coloration. Cette coloration sert à mettre en évidence les différentes cellules qui seront analysées par la suite. Cependant au fil du temps la lame se détériore, ce qui entraîne une perte d'informations. Le fait de numériser une lame permet donc de garder intactes la quantité et la qualité de l'information qu'elle contient. De plus une lame numérisée peut se copier, se faire analyser, se communiquer et se diffuser à l'infini.

La télémicroscopie a mis du temps à se développer. Ceci est principalement dû à l'inadaptation des NTIC concernant la manipulation des grands volumes de données générés par les analyses microscopiques. Néanmoins, grâce à l'évolution rapide de ces dernières années au niveau des techniques de stockage et de traitement de l'information, des solutions, qui à la base étaient trop consommatrices de ressources, peuvent voir le jour.

Dans le cadre de ma dernière année d'étude à l'Institut d'informatique des Facultés Universitaires Notre-Dame de la Paix, j'ai eu l'occasion d'effectuer un stage à la Clinique Universitaire de Mont-Godinne.

Au cours de ce stage nous avons eu l'opportunité de travailler sur un projet ayant pour objectif de contribuer à l'élaboration, déjà en cours, d'un instrument de laboratoire permettant la composition automatique de frottis numériques.

Des travaux dans le domaine de la télémicroscopie ont déjà été menés par des stagiaires à la Clinique de Mont Godinne. Il y a entre autre [Mer00] qui présente une étude sur la mise au point d'une station de télémédecine pour le laboratoire d'hématologie, et plus récemment [Geo02] qui présente la mise au point d'une solution de télémicroscopie basée sur la constitution de galeries d'images. Enfin [Zuy03] qui expose deux applications, tout d'abord un microscope virtuel et ensuite un système permettant la composition automatique de frottis numériques. Cette dernière constitue le prolongement de l'application présentée dans la partie 3 de cet ouvrage. Ce mémoire expose les recherches effectuées et les résultats obtenus durant la réalisation de ce stage.

La première partie dénommée « état de l'art » présente les notions de base de la télémédecine et de la télémicroscopie. Elle analyse ensuite les deux approches au travers desquelles les concepts de télémédecine et télémicroscopie ont été développés, à savoir l'approche « store and forward » et l'approche « real time ».

Nous définissons ensuite la notion de frottis numérique et son lien avec les méga-images ainsi que les giga-images. Nous expliquons également pourquoi la coregistration tient une place importante dans ces différentes notions, et nous finissons par exposer les différentes caractéristiques d'une image numérique.

Dans la deuxième partie, « matériel et méthode », nous décrivons les différents outils qui ont été mis à notre disposition pour la réalisation de notre stage.

Au niveau matériel il y a principalement la station d'acquisition, munie d'un microscope électronique, d'un autofocus matériel et d'une caméra vidéo. D'un point de vue

logiciel nous présentons le programme Analysis pro 3.0, la norme de compression Jpeg2000, le logiciel jj2000 ainsi que les travaux précédemment réalisés dans le cadre des stages réalisés à la Clinique universitaire de Mont Godinne dans le domaine de la télémicroscopie.

La troisième partie, « Analyse et résultats » est dédiée à la description de la solution réalisée dans le cadre du stage à savoir la réalisation de frottis numériques : application aux giga-images.

La quatrième et dernière partie, « Discussion », met en évidence les limites de l'application développée dans la partie 3, et pour chaque limite elle tente de donner des indications pour une amélioration future.

Partie 1 : Etat de l'art

Chapitre 1 : De la télémédecine à la télémicroscopie

1.1 La télémédecine

D'un point de vue **historique,** le concept de **télémédecine** est apparu dans les années 50. Celui-ci avait pour but de répondre aux problèmes d'isolement géographique de la population. Dans les années 90, l'évolution des technologies de l'information et de la communication, notamment l'amélioration de la compression audio/vidéo, l'augmentation du débit et de la fiabilité des moyens de communication ainsi que la diminution de leurs coûts ont offert à la télémédecine les moyens nécessaires pour une utilisation à grande échelle.

Il n'est pas aisé de **définir** ce qu'est la télémédecine tant celle-ci touche des domaines différents.

Suivant Montbel « la **télémédecine** consiste en l'utilisation des télécommunications et des technologies de l'information pour permettre l'accès et la prestation des soins à distance et recueillir, organiser et partager les informations cliniques requises afin d'évaluer l'état du patient, de poser un diagnostic et d'établir un traitement ». [WebTelem-1]

« Plusieurs familles d'utilisation de la télémédecine peuvent être définies, selon que l'application concernée vise principalement à :

- Echanger des d'avis entre professionnels de santé : **téléconsultation** et **téléexpertise**.
- Assister à distance, principalement par des conseils diagnostiques et thérapeutiques, un patient localement démuni : **téléassistance**.
- Surveiller à domicile une fonction vitale défaillante: **télésurveillance**.
- Pratiquer totalement et exclusivement à distance un acte médical : **télédiagnostic**, **téléchirurgie**...
- Organiser la circulation des données dans un réseau de santé : **cyber-réseaux de santé**
- Délivrer des informations voire un enseignement : **cyberformation** (ou **e-Learning**).

- Participer à la gestion des systèmes de santé : **cybermanagement** (ou **managements**), ou plus prosaïquement encore, offrir aux patients un accès direct et permanent à leur dossier de santé ou à des **téléservices médicaux** (**e-santé**).

De nombreuses applications mélangent, à des degrés divers, ces différentes familles de téléservices. » [Haz03]

1.2 La télémicroscopie

Une définition de la télémicroscopie se trouve dans [Géo02] :

« La **télémicroscopie** est la pratique de la visualisation et du diagnostic, à distance, d'images en provenance d'un microscope. Cette pratique nécessite la visualisation d'images numérisées sur écran vidéo plutôt qu'au travers des oculaires du microscope et également un moyen de communication entre le poste de visualisation et le microscope afin de permettre la transmission des images. »

Cette définition de la télémicroscopie reste assez large et ne permet pas de classer la télémicroscopie dans une seule famille de téléservices. Il est vrai que la télémicroscopie pourrait aussi bien être classée dans la famille du **télédiagnostic** que dans celle de la **téléexpertise**.

En effet, le fait de numériser l'information en provenance du microscope « permet de pratiquer totalement et exclusivement à distance un acte médical » on peut voir dans cette définition l'acte effectué par un cytologiste lorsque celui-ci réalise une analyse de frottis à distance, cet acte peut clairement être rattaché au télédiagnostic. Quant à la téléexpertise, il va de soit que si l'information est numérisée, elle est plus facile à partager. Donc si lors d'une analyse quelconque, le besoin d'un avis expert se fait sentir, il est aisé de faire parvenir à ces personnes la lame numérisée, dans le but d'avoir leur avis. Cela rencontre donc bien la définition de téléexpertise. Ces informations numérisées, peuvent aussi avoir d'autres utilisations, comme par exemple l'assurance qualité ou l'apprentissage à distance.

La télémicroscopie au même titre que la télémédecine en général s'est développée suivant deux approches : une approche dite « statique » ou « store-and-forward » et une autre dite «dynamique » ou « real time ».

1.2.1 L'approche « Statique » ou « Store and Forward »

La méthode « Store and forward » consiste à stocker l'information temporairement avant de l'acheminer plus tard vers sa destination finale. Cette méthode a comme but principal de favoriser la communication entre un analyste et un expert, l'analyste étant situé sur le site, et l'expert se trouvant à distance, la figure 1.1 illustre une telle relation.

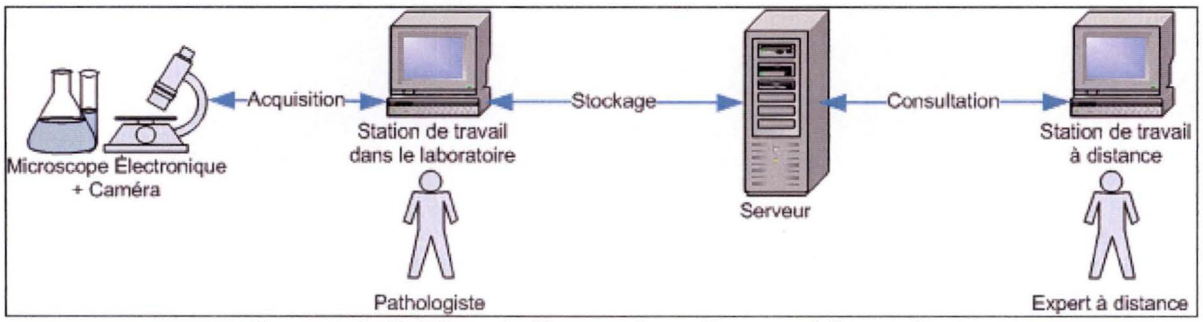


figure 1.1: Schéma de l'approche « Store and forward »

En télémicroscopie, cette solution consiste à sélectionner et acquérir un ensemble d'images représentatif du frottis et les envoyer vers une station de travail à l'aide d'un moyen de télécommunication. Une fois les phases d'acquisition et de stockage effectuées, les informations sont transmises vers la personne se trouvant à distance, le plus souvent un expert. Cette transmission peut se faire par différents moyens, les plus répandus sont l'email, le ftp[1] ou encore par la page web.

Il est clair que le transfert par fichier joint par email est le plus facile à mettre en œuvre, cependant il présente l'inconvénient de ne pouvoir acheminer que peu de données à la fois, car la plupart les serveurs de messageries limitent la taille des e-mails pour éviter le piratage. Sans oublier que tout le monde ne bénéficie pas d'une ligne à haut débit.

---

[1] En anglais FTP signifie « file transfer protocol », il s'agit d'un protocole de transfert de fichier.

Le transfert d'images en utilisant la technique ftp est plus approprié pour transférer un ensemble d'images. Les images sont stockées sur un serveur et les personnes ayant un accès à ce serveur ont la possibilité de télécharger les images précédemment stockées. Cette méthode n'est donc plus restreinte que par la capacité du serveur où sont stockées les images, ainsi que par la bande passante[2] de la personne qui les télécharge.

Le troisième moyen de transfert d'images est plus interactif que les deux premiers. Il s'agit d'une page HTML[3] contenant un ensemble de petites images représentant un frottis numérisé souvent appelé « galerie d'images ». La personne située à distance peut selon son choix faire un agrandissement de n'importe quelle image pour obtenir une meilleure définition de celle-ci. Cette méthode n'est pas encore appliquée à grande échelle car la difficulté de numériser un frottis en entier pose encore beaucoup de problèmes, tant au niveau de la capacité de stockage qu'au niveau de l'organisation des images le représentant.

Le problème du volume engendré par la numérisation complète oblige l'analyste à faire un choix lors de la phase d'acquisition pour ne numériser que les parties les plus pertinentes du frottis. Ce choix d'images peut être effectué de différentes manières : soit par une sélection au hasard, une sélection par un pathologiste expert, ou encore par une sélection automatique comme il en est question dans [Geo02]. Les images résultant de cette acquisition ne représentent donc qu'une partie de l'information contenue dans le frottis. Cette perte d'information imposée par la masse de données à gérer est l'un des principaux problèmes de la méthode dite « statique », car de ce choix d'images dépendra en grande partie les résultats de l'analyse réalisée à distance.

« Selon [Lam00], le taux de diagnostiques corrects pour les solutions « store and forward » se situe entre 85 et 90%. Bien que le taux de faux diagnostics positifs soit sensiblement plus faible que celui des faux diagnostics négatifs, ces résultats sont encore en dessous des taux atteints en microscopie conventionnelle. » [Zuy03] Par faux diagnostic positif nous entendons un diagnostique ne révélant pas une pathologie présente, et par faux

---

[2] Bande passante : Capacité maximale de débit sur une liaison donnée.
[3] HTLM : HyperText Markup Language = format de document du web.

diagnostique négatif nous voulons exprimer le fait qu'une pathologie ait été détectée alors qu'il n'y en avait pas de présente.

Les avantages principaux de la solution « store and forward » résident d'une part dans le peu de moyens qu'elle nécessite par rapport à la méthode « dynamique » explicitée ci-dessous. D'autre part, le fait que cette méthode soit asynchrone[4] offre certaines possibilités comme le stockage des informations numérisées pour une utilisation ultérieure ou encore la possibilité de faire parvenir le même échantillon numérique à plusieurs experts simultanément.

1.2.2 L'approche « Dynamique » ou « Real Time »

La méthode « **Robotic Real Time** » est caractérisée par la prise **de contrôle à distance du microscope**. En effet, l'expert ne se trouvant pas sur le site a la possibilité de contrôler les différentes fonctionnalités du microscope, et de voir l'image en temps réel sur son écran. Le contrôle se fait la plupart du temps à l'aide d'un logiciel qui reproduit les contrôles du microscope. On peut voir sur la figure 1.2 que cette solution présente l'avantage de ne pas nécessiter la présence d'une personne à proximité du microscope lors de l'analyse à distance.

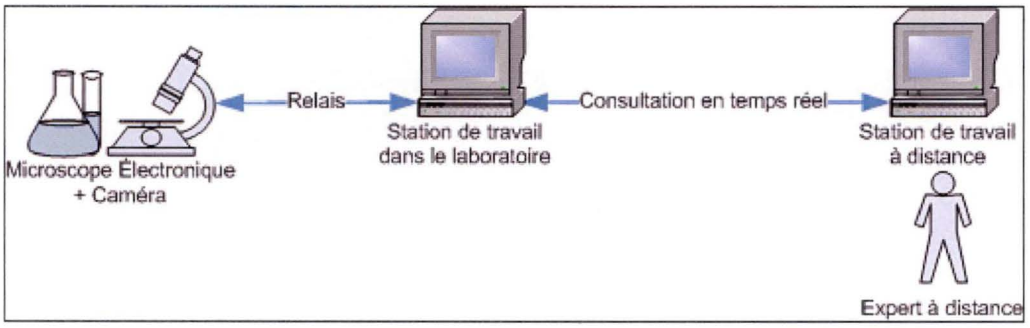


figure 1.2: Schéma de l'approche « Robotic Real Time »

[4] Asynchrone : Mode de communication dans lequel deux ordinateurs s'échangent des informations sans être synchronisés.

Dans l'approche « Real time » il existe une deuxième méthode appelée « Non-robotic Real Time ». A la différence de la méthode « Robotic » elle utilise un microscope contrôlé manuellement par un analyste se trouvant sur le site. Les images sont enregistrées par une caméra et transmise par le biais d'un logiciel de vidéo conférence[5] à un expert se trouvant à distance. L'expert qui se trouve à distance dirige la personne aux commandes du microscope, celle-ci devant effectuer les manipulations nécessaires à l'analyse du frottis.

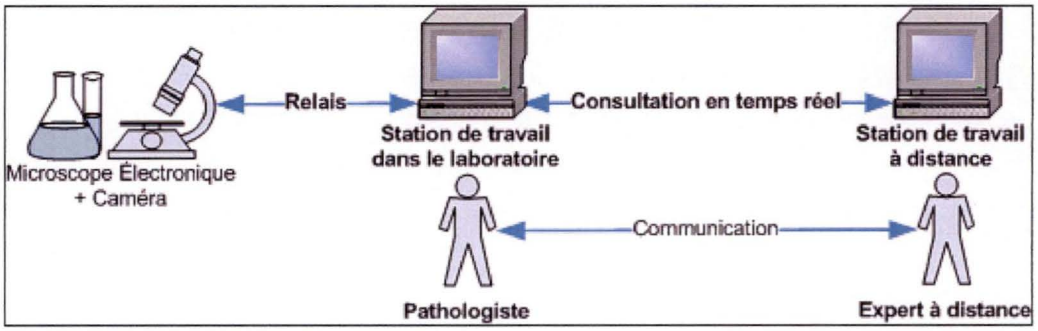


figure 1.3 : Schéma de l'approche « Non- robotic Real Time »

Les moyens matériels devant être mis en œuvre pour arriver à de telles solutions sont particulièrement importants. En effet la station de travail devrait disposer d'un microscope électronique, d'une caméra vidéo haute résolution et de moyens de télécommunication suffisants pour envoyer des images de qualité en flux continu.

Ces méthodes présentent l'avantage d'être plus objectives étant donné qu'il n'y a pas de phase d'acquisition, celle-ci étant à la base de la différence entre les résultats obtenus par une méthode « store and forward » et les méthodes dites traditionnelles. Car il est évident que la sélection effectuée lors de cette phase d'acquisition réduit fortement la quantité d'informations.

Etant donné que notre solution de composition automatique de frottis numériques nécessite plusieurs phases (stockage, coregistration,...) qui ne peuvent se faire en temps réel, c'est l'approche « Store and forward » qui a été choisie pour réaliser notre application.

---

[5] NetMeeting par exemple.

Chapitre 2 : Elaboration d'un frottis numérique

2.1 Le frottis numérique

Dans cet ouvrage nous considérerons qu'un frottis numérique est la représentation sous format électronique d'un frottis dit « traditionnel », la figure 2.1 illustre les différents éléments intervenant lors de la réalisation d'un frottis numérique.

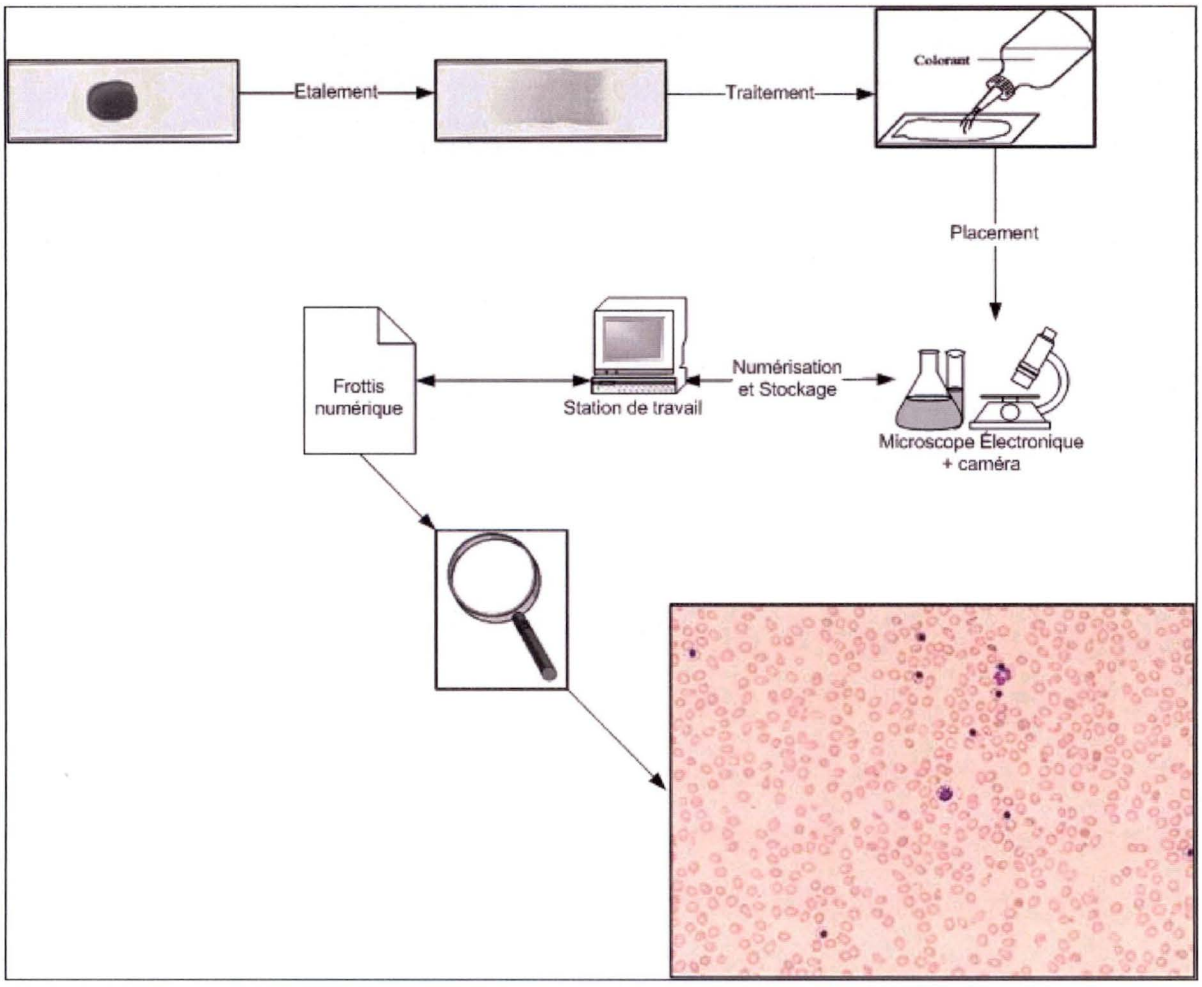


figure 2.1 : Les étapes de construction d'un frottis numérique

Les frottis numériques résultant de la télémicroscopie, sont destinés à différentes utilisations et ont, par rapport aux frottis traditionnels, certains avantages. En effet, les frottis numériques permettent l'analyse asynchrone réalisée par un expert, comme il a été vu dans la partie « store and forward ». Auparavant la demande d'une expertise à distance par un hôpital n'ayant pas de moyen d'analyse à sa disposition, se faisait par l'envoi de lames via des moyens de transports traditionnels[6]. Il est donc évident que la dématérialisation des frottis traditionnels au profit des frottis numériques améliore la vitesse de communication entre cytologistes distants l'un de l'autre.

Les frottis numériques peuvent aussi faciliter le contrôle de qualité réalisé dans les laboratoires d'analyse cytologique. Ils ont même constitué une révolution dans ce domaine car auparavant, les contrôles de qualité se faisaient sur base d'échantillons différents d'un même prélèvement. Chaque laboratoire recevait donc un échantillon différent sensé représenter la même information. Ces laboratoires devaient par la suite analyser ses échantillons et faire parvenir les résultats à l'organisme responsable du contrôle. Le problème se situait dans le fait que les différents laboratoires ne recevaient pas la même base à analyser. Avec l'arrivée des frottis numériques, ce problème ne se pose plus car tous les laboratoires reçoivent le même frottis. Les frottis numériques utilisés par le contrôle de qualité, sont des frottis « grand champ ». Ils ont cette appellation car, pour procéder à une analyse complète d'une lame, il n'est pas nécessaire d'avoir un frottis entier, mais il faut une partie longitudinale du frottis d'origine. C'est-à-dire un échantillon de toute la longueur du frottis et d'une largeur suffisante pour recueillir l'ensemble des différentes cellules contenues sur la lame.

Un autre avantage des frottis numériques, se situe dans le fait que lors de la réalisation d'un frottis, certains colorants sont utilisés pour mettre en évidence les éléments à observer. On peut donc considérer qu'un frottis non numérisé perd de l'information et de la qualité au cours du temps. Un bon moyen de garder ces informations consiste donc à numériser ce frottis.

Certains problèmes inhérent à la réalisation de frottis numériques ont été mis en exergue et solutionnés dans [Zuy03]. Ces problèmes étaient principalement dûs à la précision

---

[6] La poste ou autre.

du microscope, le fonctionnement de l'autofocus, la quantité d'informations à stocker et la manipulation des grandes quantités d'informations.

La digitalisation complète d'une lame à fort grossissement pose encore un gros problème au niveau du volume de données à manipuler. En effet, deux facteurs sont à l'origine de l'importante place mémoire prise par un frottis numérique : le premier est « la taille de la zone numérisée sur la lame » et le second est « le grossissement auquel les images sont capturées ». Ceci étant, les frottis numériques permettant aux cytologistes de pouvoir analyser une lame dans les meilleures conditions, doivent être réalisés avec un objectif de grossissement 100x. Nous verrons avec le tableau suivant l'importance de ce paramètre. Si l'on considère que « le grossissement auquel les images sont capturées » est le 100x, le seul facteur qui reste variable est « la taille de la zone numérisée sur la lame » ou autrement dit « le nombre d'images numérisées » pendant la phase d'acquisition.

[Zuy03] Les résultats du tableau 1.1 donnent une estimation du volume en giga Bytes que représente un « frottis numérisé » à différents grossissements. Ces chiffres ont été établis sur base de mesures effectuées avec le microscope du laboratoire de la Clinique Universitaire de Mont-Godinne et la taille de 20x25 mm correspond à une estimation de la zone entière d'étalement d'un frottis sanguin sur une lame.

Tableau 2.1 – Estimation du volume nécessaire à la digitalisation d'une zone de 20x25mm

Grossissement	Taille de 1 pixel numérisé (en µm)	Nombre de pixels nécessaires	Volume nécessaire pour des images RGB (en GB)
40 x	0.204	98040 x 122550	33,6
60 x	0.13	153847 x 192307	82,6
100 x	0.08	250000 x 312500	218,3

Pour bien comprendre d'où vienne ces chiffres, il faut comprendre comment une image numérisée est représentée en mémoire. Cela est expliqué dans le chapitre 3.

2.2 De la méga image à la giga-image

Comme nous l'avons vu dans le tableau précédent, le volume de données à manipuler est énorme. Nous avons donc décidé d'appeler « méga-image » la représentation d'une partie de frottis numérique, et « giga-image » un ensemble de méga-images.

« Dans la littérature, lorsque l'on parle de méga-image, on fait référence à des images dont les dimensions sont gigantesques, c'est-à-dire, des images qui nécessitent une grande quantité de ressources pour leur stockage et qui entraînent des délais d'accès importants. »[Zuy03]

La figure 2.1 illustre la relation qui existe entre les images acquises par le microscope électronique, les méga-images résultant de la coregistration de ces images acquises précédemment, et les giga-images résultat de la coregistration des différentes méga-images.

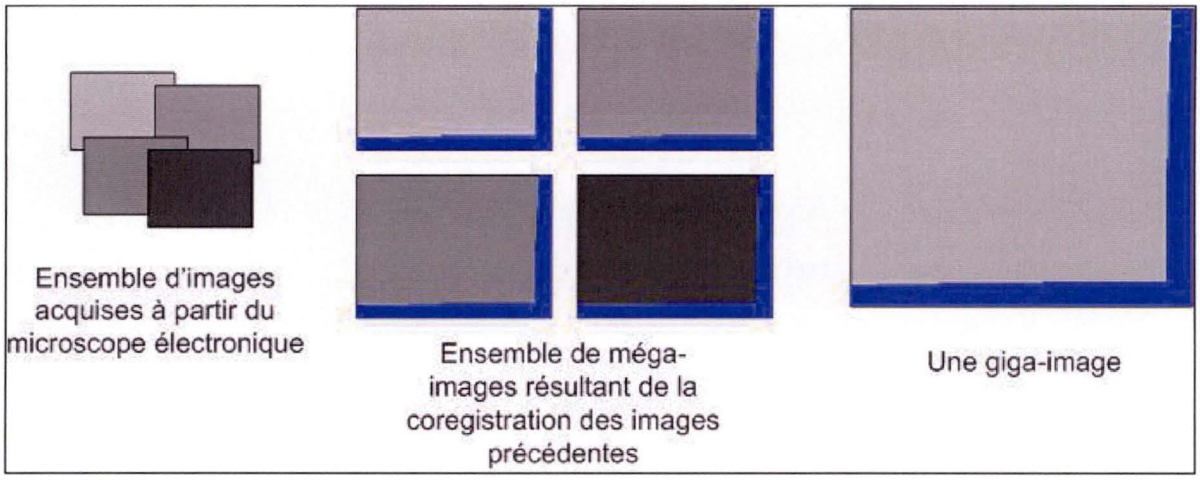


figure 2.2 : Relation entre image, méga-image et giga- image

Dans cet ouvrage, nous considérons que la différence entre méga-image et giga-image se situe principalement au niveau de dimension maximale de l'une et de l'autre. Cette dimension est fixée par la façon dont une méga-image et une giga-image sont construites. La méga-image est construite suite à la coregistration en Ram d'un ensemble d'images en provenance du microscope. La taille d'une méga-image est donc limitée par la taille de la mémoire Ram. Une giga-image est, quant à elle, construite en coregistrant un ensemble de

méga-images et cette dernière coregistration se fait en utilisant le disque dur. La dimension maximale d'une giga-image peut donc être supérieure à celle d'une méga-image.

2.3 L'importance de la coregistration

« On entend par coregistration le processus d'alignement ou de mise en correspondance de deux images, appelées « image maître » et « image esclave » recouvertes entièrement ou partiellement par une même zone. Cette zone sous-entend donc que les images représentent entièrement ou partiellement une même information que l'on appellera la « scène observée » ». [Zuy03]

Dans le domaine de la télémicroscopie, la coregistration prend une place importante. En effet, lors de la capture d'une image avec la caméra CCD, une telle capture est appelée *snapshot*, la zone numérisée par celle-ci est relativement petite, chaque *snapshot* couvre une zone de 765x573 pixels ce qui fait un total de 438 .345 pixels par *snapshot*. Si l'on tient compte du tableau 2.1 il faudrait numériser 78 125 000 000 pixels pour arriver à numériser une lame complète il faudrait 78 125 000 000/438.345 = environ 178.227 *snapshots*. La coregistration est indispensable pour permettre un agencement correct de ces *snapshots* entre eux.

Certains problèmes sont présents lors de la réalisation de giga-images. En effet pour réaliser une giga-image résultant de la coregistration de plusieurs méga-images, il faut inexorablement passer par le disque dur, car la mémoire centrale est insuffisante. Ce passage par le disque dur et la coregistration en résultant seront explicités dans la partie 3 : Analyse et Résultats.

Chapitre 3 : Le point de vue technologique

3.1 La technologie des réseaux et Internet

Comme nous l'avons vu précédemment Internet joue un rôle de plus en plus important dans le domaine de la télémédecine. En effet l'évolution rapide des technologies de l'information et de la communication a propulsé la télémédecine sur le devant de la scène. Ces dernières années ont vu une explosion du domaine des télécommunications. La démocratisation des offres d'accès Internet à haut débit par le biais des technologies telles que l'ADSL[7] ou le câble ont permis à un nombre croissant de personnes d'accéder au NET.

D'autres évolutions ont aussi joué un rôle dans l'évolution de la télémédecine. Pour en citer quelques uns il y a par exemple, l'augmentation sans cesse croissante de la vitesse des ordinateurs, l'accroissement de la capacité de stockages de ceux-ci…

3.2 La compression d'images

Les volumes de données à manipuler aux travers des méga images sont énormes. Il va donc de soi qu'une technique de compression d'images sera utilisée dans le but de réduire ce volume.

Les méthodes de compression utilisées dans les domaines médicaux, sont généralement des méthodes « **sans perte** », car il est normal que les spécialistes puissent disposer de toute l'information disponible sans que des artéfacts aient été introduits par l'utilisation de techniques de compression /décompression inappropriées. Néanmoins, si le

[7] ADSL : Asymmetrical Digital Subscriber Line

volume des images à manipuler est trop important il est nécessaire de faire appel aux techniques de compression « **avec perte**. »

La compression sans perte, également appelée « *lossless* », est une méthode de compression de données qui restitue, après la décompression, un fichier numérique identique à l'original. De telles méthodes sont utilisées dans les formats tel que GIF ou JPEG2000.

Cependant, ce type de méthode de compression ne permet pas toujours d'obtenir des taux de compression importants, surtout lorsque l'image ne présente que peu de répétition de couleurs, qui sont à la base de la plupart des méthodes de compression sans perte. Des techniques de compression avec perte ont donc été développées. On les nomme ainsi car l'information « numérique » de départ est perdue lors de la compression : il est impossible d'obtenir une copie exacte de l'image de départ à partir de l'image compressée. Néanmoins, l'œil humain ne peut, lorsque les pertes ne sont pas trop importantes, faire la différence avec l'image d'origine. L'intérêt de ce type de méthode est de fournir des taux de compression beaucoup plus importants et est donc beaucoup utilisée dans des domaines tels que la photographie numérique où les images manipulées peuvent être de taille très importante. Le format le plus couramment utilisé pour la compression avec perte est le JPEG.

3.3 L'image numérique

Les informations suivantes sont issues du mémoire réalisé par C. Kadour [Kad99]

3.3.1 Définition [Gon77] :

L'image numérique est une image dont la surface est divisée en éléments de taille fixe appelés cellules ou pixels, ayant chacun comme caractéristique un niveau de gris ou de couleurs prélevé à l'emplacement correspondant dans l'image réelle, ou calculé à partir d'une description interne de la scène à représenter.

La numérisation d'une image peut être définie comme la conversion de celle-ci de son état analogique (distribution continue d'intensités lumineuses dans un plan xOy) en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques f(x, y) où :

- x, y : coordonnées cartésiennes d'un point de l'image.
- f(x, y) : niveau de gris en ce point.

3.3.2 Le pixel [Enc97] :

Le pixel est le plus petit point de l'image, c'est une entité calculable qui peut recevoir une structure et une quantification. Si le bit est la plus petite unité d'information que peut traiter un ordinateur, le pixel est le plus petit élément que peuvent manipuler les matériels et logiciels d'affichage ou d'impression. La lettre A, par exemple, peut être affichée comme un groupe de pixels dans la figure ci-dessous :

La quantité d'information que véhicule chaque pixel donne des nuances entre images monochromes et images couleurs. Dans le cas d'une image monochrome, chaque pixel est codé sur un octet, et la taille mémoire nécessaire pour afficher une telle image est directement liée à la taille de l'image.

Dans une image couleur (R.V.B.), un pixel peut être représenté sur trois octets : un octet pour chacune des couleurs : rouge (R), vert (V) et bleu (B).

Il faut aussi savoir que plus il y a de pixels dans une image numérique, plus celle-ci sera une représentation proche de l'image originale. Le nombre de pixels dans une image est aussi appelé la « résolution de cette image ».

3.3.3 Les images en niveaux de gris [Har95]:

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de

niveaux intermédiaires. Donc, pour représenter les images à niveaux de gris, on peut attribuer à chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être comprise par exemple entre 0 et 255. Chaque pixel n'est donc plus représenté par un bit, mais par un octet. Pour cela, il faut que le matériel utilisé pour afficher l'image soit capable de produire les différents niveaux de gris correspondant. Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la " couleur " de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux.

3.3.4 Les images en couleurs [har95] :

La représentation des couleurs s'effectue de la même manière que les images « à niveau de gris » avec cependant quelques particularités. En effet, il faut tout d'abord choisir un modèle de représentation. On peut représenter les couleurs à l'aide de leurs composantes primaires. Les systèmes émettant de la lumière (écrans d'ordinateurs,...) sont basés sur le principe de la synthèse additive : les couleurs sont composées d'un mélange de rouge, vert et bleu (le modèle R.V.B.). La représentation en couleurs réelles consiste, elle, à utiliser 24 bits pour chaque point de l'image. Huit bits sont employés pour décrire la composante rouge (R), huit pour le vert (V) et huit pour le bleu (B). Il est ainsi possible de représenter environ 16,7 millions de couleurs différentes simultanément. Cela est cependant théorique, car aucun écran n'est capable d'afficher 16 millions de points. Dans la plus haute résolution (1600 x 1200), l'écran n'affiche que 1 920 000 points. Par ailleurs, l'œil humain n'est pas capable de distinguer autant de couleurs.

3.3.5 La dimension [Had97] :

La dimension d'une image est la taille de celle ci. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image.

3.3.6 La résolution [Enc97] :

La résolution d'une image est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). On utilise aussi le mot « résolution » pour désigner le nombre total de pixels affichables horizontalement ou verticalement sur un moniteur. Plus grand est ce nombre, meilleure est la résolution.

3.3.7 La luminance [Tab96] :

La luminance est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface, pour un observateur lointain. Le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet.

3.3.8 Le contraste [Tab96] :

Le contraste est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images.

Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L1 - L2}{L1 + L2}$$

3.3.9 Le poids d'une image [WebCcm] :

Pour connaître le poids (en octet) d'une image il faut compter le nombre de pixels que contient l'image. Cela revient à calculer le nombre de cases du tableau, soit la hauteur de celui-ci que multiplie sa largeur. Le poids de l'image est alors égal à son nombre de pixels

que multiplie le poids de chacun de ses éléments. Par exemple pour une image d'une dimension de 640x480 pixels en couleur vraie, c'est-à-dire un pixel codé sur 3 octets, un pour le rouge un pour le vert et un pour le bleu. Le poids d'une image se calcul comme ceci :
On calcul le nombre de pixels de l'image : 640*480 = 307200 pixels
On calcul ensuite le poids de cette image : 307200*3 =921600 octets
On traduit ensuite le nombre obtenu en octet pour l'avoir kilo octet : 921600/1024 = 900Ko

Dans le cas du tableau repris dans le chapitre un, le calcul est le même. Pour la zone à numériser au zoom 100x, la taille de l'image est de 250000 x 312500 = 78 125 000 000 pixels, si on multiplie par 3 cela donne 234 375 000 000 octet. Pour exprimer ce nombre en giga octets il faut le diviser par 1024^3 ce qui donne 218,2787 Go. Ce qui correspond bien au volume indiqué dans le tableau.

Partie 2 : Matériel et méthode

Chapitre 4 : L'infrastructure disponible

Différents composants tant matériels que logiciels étaient à notre disposition pour la réalisation des applications présentées dans la partie 3 «Analyse et résultats ». Nous disposions principalement d'une station d'acquisition illustrée ci-dessous ainsi que des programmes installés sur l'ordinateur relié au microscope.

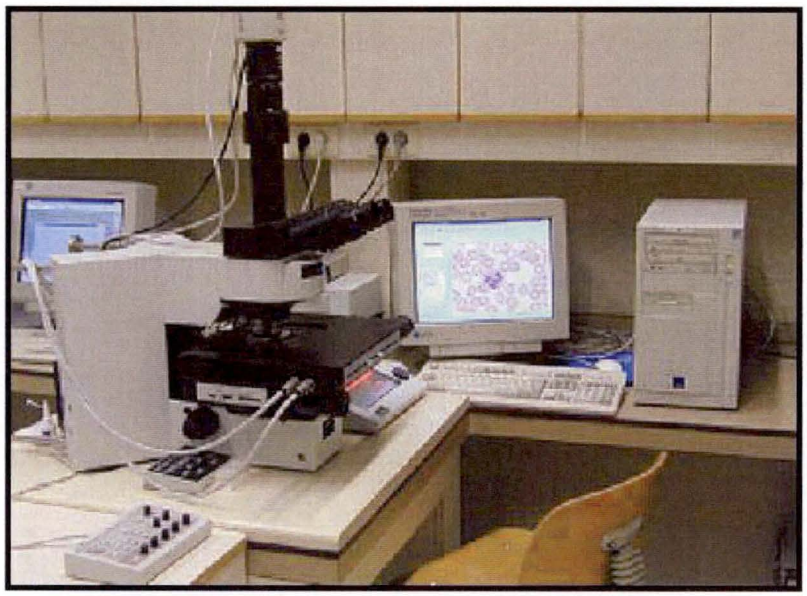


figure 4.1 : Station d'acquisition de Mont Godinne.

4.1 La station d'acquisition :

L'élément principal de la station d'acquisition est le microscope électronique « AX70 » de la marque Olympus illustré par la figure 4.2

Les différents composants de la station d'acquisition sont :

- Une platine motorisée. Par platine on doit comprendre l'endroit ou les lames sont disposées avant d'être visionnées au travers du microscope. Cette platine peut se mouvoir sur les Axes Y, X et Z, l'axe Z étant géré soit par l'utilisateur soit par l'autofocus.

- Un boîtier U-MCB[8] permettant le réglage de certaines données comme les filtres de couleurs ou encore la luminosité.

- Une caméra analogique Sony « DXC-950 » permettant l'acquisition de 3 images CCD représentée par la figure 4.3

- Un autofocus hardware « U-AF ».

La platine motorisée, le boîtier U-MCB ainsi que l'autofocus hardware sont tous les trois reliés à l'ordinateur de la station de travail par le biais d'un câble série respectant la norme « RS232 ». Ceci permet donc une automatisation quasi complète de tous les éléments. L'ordinateur de cette station est doté d'une capacité de stockage de 60 Go, d'un processeur de 730 Mhz ainsi que de 640 Mo de mémoire RAM. Le système d'exploitation installé sur la machine est « Microsoft Windows NT 4.0 » et la machine est connectée en permanence au réseau local et possède un accès à Internet.

---

[8] Unit-Main Control Board.

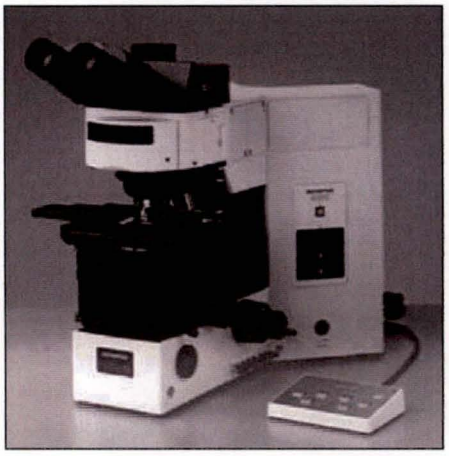


figure 4.2 Microscope « AX70 »
de chez Olympus.

figure 4.3 Caméra analogique
« DXC-950 » de chez Sony.

4.2 Le logiciel Analysis

L'ordinateur faisant partie de la station d'acquisition dispose du programme « Analysis Pro 3.0 ». Ce logiciel est conçu pour l'acquisition, l'analyse, le traitement et l'archivage d'images. Il présente plusieurs avantages. D'une part il est doté d'un module comprenant des librairies permettant de gérer les différentes fonctionnalités du microscope « AX70 » d'Olympus. D'autre part il possède un composant permettant à l'utilisateur de personnaliser l'application en créant lui-même ses modules personnels. Le langage utilisé par ce composant est un langage proche du ANSI-C appelé « Imagin-C » et qui a comme particularité de proposer certaines fonctions supplémentaires liées au traitement d'images, ainsi que de proposer un interpréteur intégré. Imagin-C est également compatible avec le « MS Windows Sorfware Developement Kit » (SDK) ce qui permet la programmation de nouvelles interfaces graphiques dans l'environnement « MS Windows ». Le logiciel Analysis permet aussi une intégration facile des nouvelles fonctions dans son interface, en y associant des boutons.

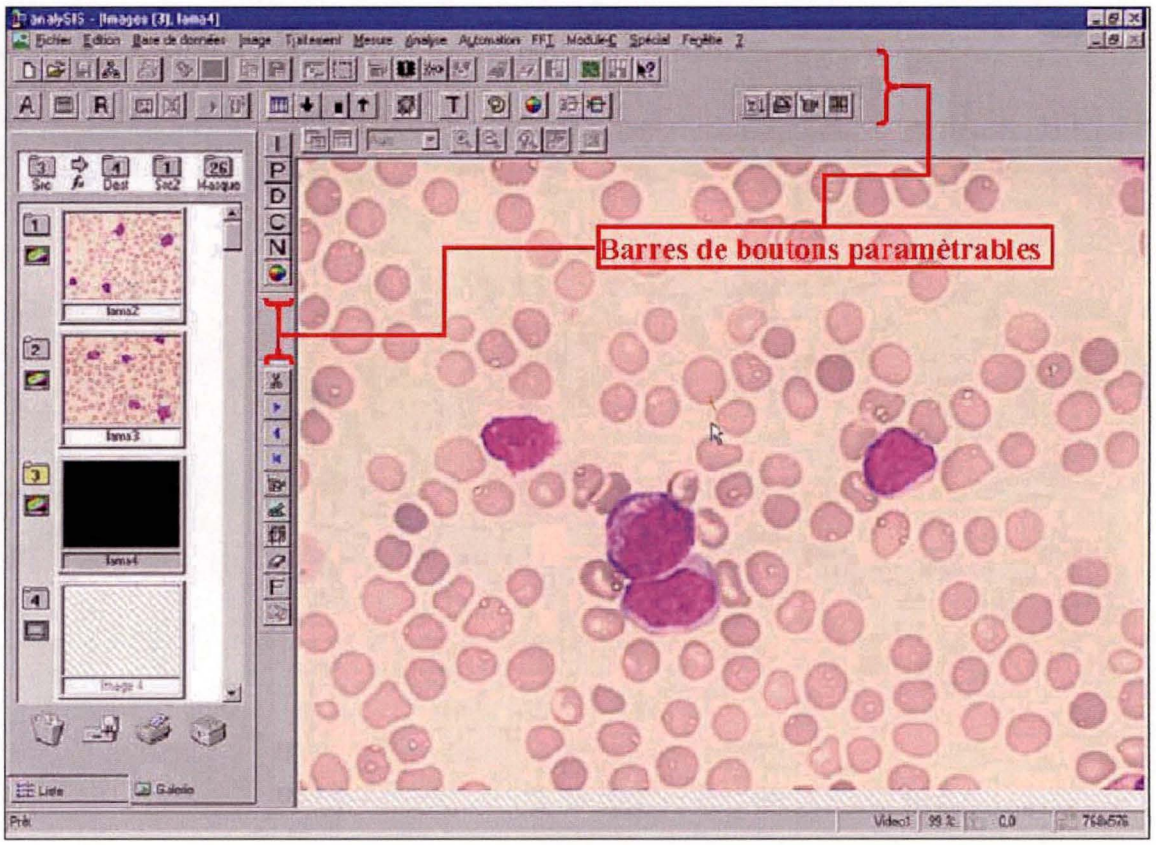


figure 4.4 : Aperçu de l'interface du logiciel Analysis Pro 3.0

4.3 Les travaux précédents

4.3.1 La réalisation automatique de galeries d'images

L'ouvrage réalisé par Benoît Georges dans [Geo02] dénommé « La télémicroscopie en cytologie hématologique » et en particulier les parties concernant « l'analyse automatique » et la « réalisation d'une galerie d'image » furent une bonne introduction pour la réalisation des applications de la partie 3. Cet ouvrage représente un bon exemple d'une application télémicroscopique de type « Store and forward ». Un autre point intéressant de cette application se trouve dans l'utilisation assez poussée du logiciel Analysis, et plus particulièrement le composant « Imagin-C » qui est utilisé en adéquation avec le module « AX70 ».

4.3.2 La composition automatique de frottis numériques

Les travaux réalisés par Louis Zuyderhoff dans [Zuy03] ont constitué la base des applications exposées dans la partie 3 : Analyse et résultats. La composition automatique de frottis numériques peut être décomposée en plusieurs phases :

1. Préparation de la lame.
2. Acquisition des images en provenance du microscope électronique.
3. Coregistration des images en mémoire.
4. Organisation et stockage des images.

Nous allons passer en revue les différentes phases,

1. Préparation de la lame :

Cette phase n'est pas du ressort de l'informaticien, mais bien du pathologiste. Dans le cas d'un frottis sanguin, un échantillon de sang est étalé sur une lame. Une fois l'échantillon de sang étalé celui-ci est coloré pour mettre en évidence les différentes cellules constitutives du frottis. La lame est ensuite placée sur le porte-lame se trouvant sur la platine motorisée.

2. L'acquisition des images en provenance du microscope électronique :

Cette deuxième phase est consécutive à la première. Une fois que l'utilisateur a choisi le nombre d'images en abscisse et le nombre d'images en ordonnée, on capture un ensemble d'images. Les images une fois capturées forment une matrice à deux dimensions comme celle de la figure 4.5. Les images qui sont capturées se recouvrent horizontalement et verticalement sur une zone de 100 pixels^2. Ce qui permet la coregistration lors de la phase suivante.

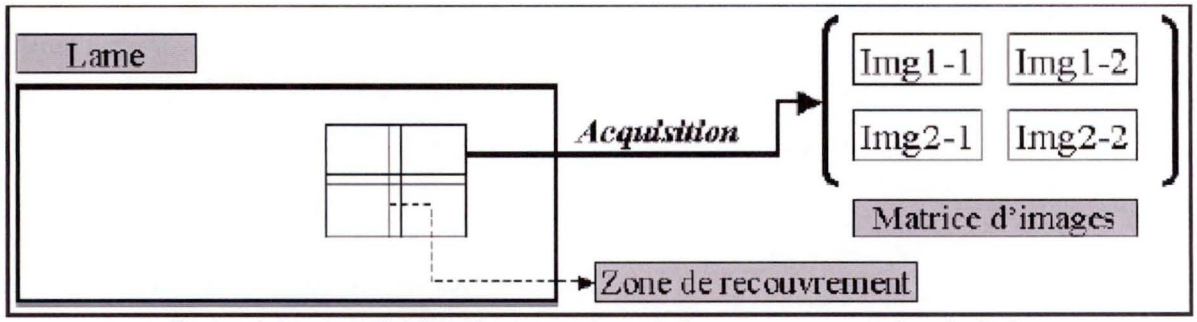


figure 4.5 : Acquisition de quatre image formant une matrice de 2x2.

3. La coregistration des images en mémoire Ram

Pour numériser une zone suffisamment grande il faut acquérir un nombre suffisant d'images et les assembler. De plus durant la phase d'acquisition des images, certaines erreurs dans l'alignement de chaque image par rapport à la position souhaitée apparaissent. Une méthode de coregistration s'est donc avérée nécessaire.

Pour faciliter la coregistration plusieurs transformations sont effectuées : réduction en échelle de gris, normalisation, post optimisation horizontale et verticale.

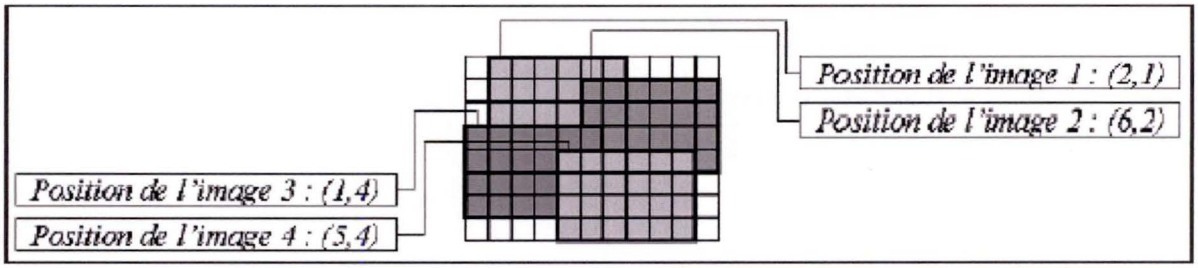


figure 4.6 : L'assemblage de quatre images en une méga-image

L'ensemble des images une fois coregistrées forme une grande image appelée méga-image. On peut voir sur la méga-image illustrée par la figure 4.7 que les bords de celle-ci ne sont pas continus. En effet, lors de la phase de coregistration, les différentes images constituant la méga-image ont été décalées pour s'ajuster les par rapport aux autres. Suite à ces ajustements, la forme de la méga-image n'est pas rectangulaire. Pour ne pas travailler avec des méga-images ayant des dimensions et formes non régulières nous avons complété les bords de la méga-image pour lui faire atteindre une dimension rectangulaire[9], comme celle de la figure 4.8.

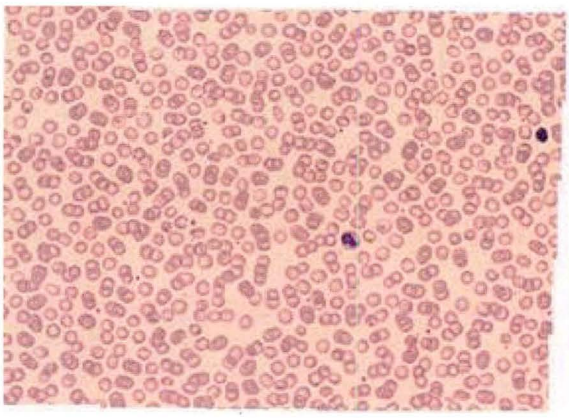


figure 4.7 : Méga-image sans remplissage

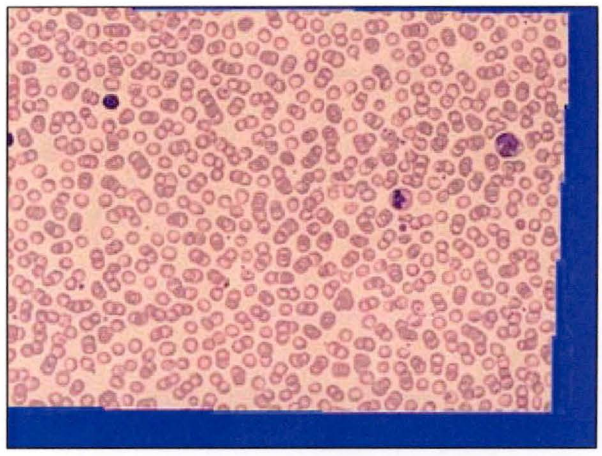


figure 4.8 : Méga-image avec remplissage

[9] Cette dimension n'est pas due au hasard on verra dans le chapitre 3 qu'elle est multiple des *tiles*. La notion de *tiles* sera expliquée dans le chapitre suivant.

4. Organisation et Stockage des images.

Les images sont acquises lors de la phase d'acquisition sous forme de fichier .TIFF, elles sont ensuite chargées en mémoire, puis coregistrées. Cette matrice d'images coregistrées appelée méga-image, est ensuite compressée et stockée sur le disque dur sous le format Jpeg2000.

La composition automatique de frottis numériques expliquée ci-dessus constitue la base de la réalisation des « frottis grands champs » ainsi que la base des applications réalisées dans la partie 3 analyse et résultats.

Chapitre 5 : La méthode de compression Jpeg 2000

5.1 Introduction à Jpeg2000

Beaucoup d'efforts ont été fournis ces dernières années pour développer une méthode de compression d'images regroupant d'un côté un large panel de fonctionnalités et d'un autre un excellent taux de compression. Cependant Jpeg2000 n'as pas été conçu dans le but de remplacer le format Jpeg, mais bien celui de pallier certains manques de celui-ci. Dans notre cas nous soulignerons particulièrement les solutions que Jpeg2000 apporte dans la navigation d'images.

5.2 L'utilité de Jpeg2000 en Imagerie Médicale

Comme nous avons pu le constater dans les chapitres précédents, la taille des images à traiter dans le domaine de la cytologie numérique pose un réel problème. En effet, les frottis numériques, pour être correctement analysés par un cytologiste, doivent d'une part représenter une surface suffisamment grande du frottis originel, approximativement $1cm^2$, et d'autre part l'objectif utilisé lors de l'acquisition doit avoir un grossissement suffisamment élevé - environ un zoom 100x - pour conférer à l'image numérique une définition suffisante. Ce frottis numérique représente donc une masse d'information importante ce qui rend l'utilisation de méthodes de compression impérative.

L'algorithme de compression Jpeg2000 présente plusieurs fonctionnalités fort intéressantes pour le milieu médical et notamment pour la manipulation des giga-images qui nous concerne directement. Car outre un bon taux de compression sans perte de qualité, ainsi qu'un excellent taux de compression avec perte, Jpeg2000 permet le codage et décodage progressif ainsi qu'un accès aléatoire aux données, ce qui, dans le cadre des giga-images, représente un énorme avantage lors du visionnage. Ces différentes fonctionnalités de Jpeg2000 seront présentées dans les points suivants.

5.3 La norme Jpeg2000

La norme Jpeg2000[10] ne définit que l'algorithme de décodage et le format des données compressées. Ceci laisse donc une grande liberté quant au choix du système de codage. La norme Jpeg2000 est constituée de plusieurs parties que nous allons présenter ci-dessous :

- **Partie 1 : Le noyau du système de codage Jpeg2000**

Comme son nom l'indique, la partie 1 de la norme définit le noyau de Jpeg2000 à savoir la technologie minimale de l'algorithme de décodage ainsi que la syntaxe du *codestream*[11] devant être comprise par tous les produits compatibles à la norme. La partie 1 introduit aussi le format de fichier spécifique dont l'extension est .jp2, celui-ci donnant la possibilité d'inclure des informations additionnelles au *codestream* du fichier.

- **Partie 2 : Extensions de Jpeg2000**

La partie 2 définit plusieurs extensions à la partie 1 afin d'augmenter le nombre de fonctionnalités ou encore d'améliorer les performances. C'est dans cette partie qu'on définit les possibilités dans le domaine des métas donnés.

- **Partie 3 : Motion Jpeg2000**
- **Partie 4 : Règle de conformité à Jpeg2000**
- **Partie 5 : Logiciels de référence**

La partie 5 comprend deux packages de codes sources qui implémentent la partie 1 de la norme Jpeg2000, à savoir d'une part JJ2000 codée en java, dans le cadre d'une collaboration entre Canon Research France, Ericsson et l' Ecole polytechnique fédérale de Lausanne. Et d'autre part le package JasPer écrit en C.

---

[10] Cette norme est reprise dans le document : « ISO/IEC 15444-1 »
[11] Traduction anglaise de « flux de données compressées »

- **Partie 6 : Format de fichier d'images composées**

Cette partie de la norme spécifie un format de fichier pour les images composées, c'est-à-dire contenant différents types de données comme par exemple, du texte, des photographies, des graphiques.

- **Partie 7 : à été abandonnée**
- **Partie 8 : « Sécurité de Jpeg2000 »**
- **Partie 9 : « Protocole d'interactivité Jpeg2000 »**
- **Partie 10 : « Jpeg2000 3D »**
- **Partie 11 : « Jpeg2000 sans fil »**
- **Partie 12 : « Le format de fichier ISO Base média »**

Aujourd'hui seule la partie 1 est publiée en tant que norme internationale. Les parties 2 à 6 sont sur le point d'être terminées et publiées et les quatre dernières parties (8-11) sont en cours de développement.

Les différents points que nous allons aborder par la suite seront directement et essentiellement en rapport avec la partie 1 de la norme Jpeg2000 car celle-ci est publiée et présente la base de la compression Jpeg2000.

5.4 Les différentes fonctionnalités de Jpeg2000

La méthode de compression Jpeg2000 présente plusieurs avantages par rapport aux méthodes de compressions traditionnelles. Certaines d'entre elles sont énumérées ci dessous :

- **Le décodage progressif :**

Le *codestream* résultant de compression par l'algorithme Jpeg2000 est à résolution multiple, c'est-à-dire que l'image est compressée résolution par résolution sans redondance. Ceci permet notamment l'utilisation de la résolution la plus appropriée au système d'affichage. D'autres agencements du *codestream* sont aussi possibles, pour obtenir un

décodage progressif en qualité comme celui de la figure 5.1, ou encore grouper l'information par rapport à sa position dans l'image. Un exemple de décodage progressif par résolution est proposé à la figure 5.2

figure 5.1 : Représentation d'un décodage progressif en qualité

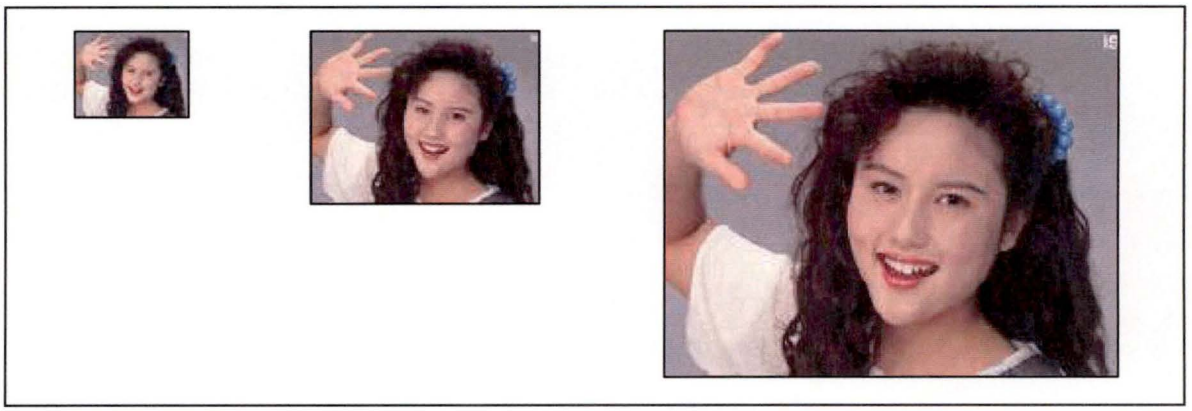


figure 5.2 : Représentation d'un décodage progressif par résolution

- **Le codage avec ou sans perte :**

Il faut souligner ici que le même algorithme permet à la fois la compression avec perte et sans perte.

- **L'utilisation de régions d'intérêts :**

Lors du codage, il est possible d'affecter à certaines parties de l'image des qualités différentes. Il est même possible de coder une partie de l'image avec perte et une autre partie sans perte.

- **L'accès aléatoires aux différentes zones de l'image :**

Il est possible de décoder certaines zones d'une image si celle-ci est grande. On imagine très bien l'utilité de cette fonctionnalité dans le contexte des giga-images.

Voici encore quelques fonctionnalités apportées par Jpeg2000 : une architecture ouverte, la possibilité d'insérer des méta données, la gestion des images 32bits, une amélioration de la protection des images.

5.5 L'algorithme de codage

Comme évoqué précédemment, seul l'algorithme de décodage est décrit dans la partie 1 de la norme Jpeg2000. Il est cependant possible de présenter un algorithme reprenant les différentes phases typiques de codage. La figure 5.3 représente le schéma d'un algorithme de codage Jpeg2000. Les phases nécessaires sont entourées d'un trait continu, et les phases facultatives sont entourées d'un trait discontinu.

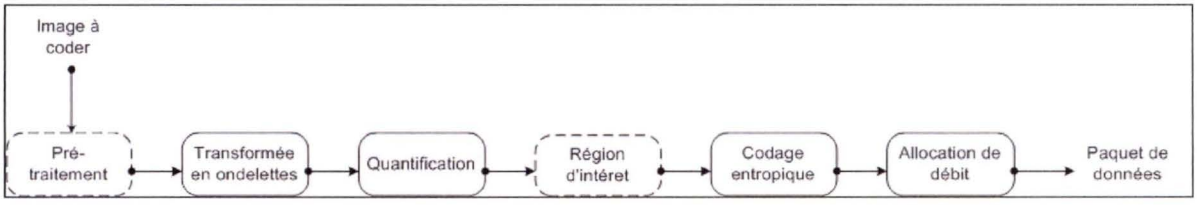


figure 5.3 : Schéma typique d'un Jpeg2000

- **La phase de pré-traitement :**

La phase de pré-traitement est composée de plusieurs étapes. Nous allons par la suite nous intéresser plus particulièrement à la découpe en *tile*[12], ainsi qu'à la « transformée couleur »[13].

La transformée couleur est optionnelle mais peut s'avérer intéressante dans le cas ou l'image à coder est du type (R, V, B)[14]. Cette transformée permet d'obtenir une représentation de l'image dans un espace de luminance/chrominance plus adaptée à la compression des données. [San01] & [WebCmm]

La découpe en *tiles* quant à elle consiste à découper l'image d'origine en éléments rectangulaires disjoints appelés « *tiles* ». Les *tiles* seront par la suite compressés de façon indépendante comme s'il s'agissait d'images différentes. Tous les *tiles* de l'image source sont de la même dimension à l'exception des bords de droite et du bas. Comme cette phase est facultative, il est possible que l'image source ne soit composée que d'un seul et unique *tile*. Les avantages de la découpe en *tiles* sont multiples. Outre un meilleur taux de compression dans certains cas, le *tiling* permet le codage et donc le décodage par région d'intérêt et donne aussi la possibilité de décoder l'image morceaux par morceaux. Nous verrons dans le chapitre 4 de la troisième partie, analyse et résultats, l'importance prépondérante que prend la phase de *tiling*.

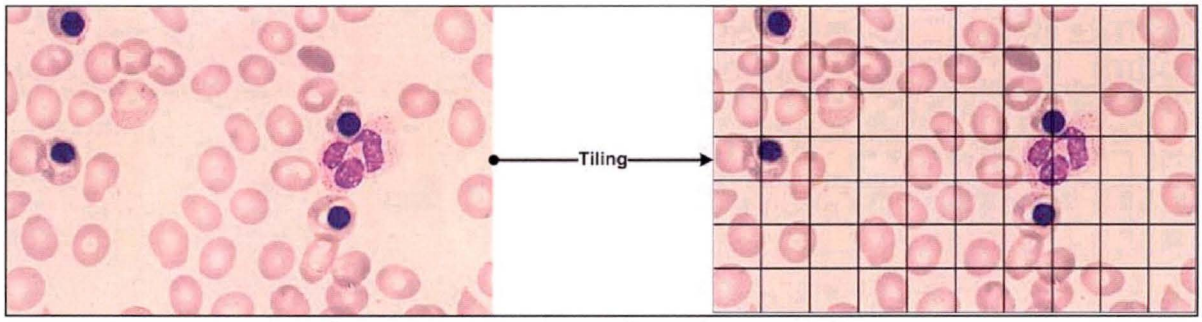


figure 5.4 : Découpage d'une image en tiles.

[12] Traduction anglaise de « tuile », « carreau ».
[13] Une explication du passages de l'espace RGB à celui de luminance /chrominance est présenté dans l'annexe 1.
[14] Ce sont les trois couleurs de base, le rouge, le vert et le bleu

- **La transformée en ondelettes discrètes**

La transformée en ondelettes discrètes à pour but d'une part d'obtenir une structure plus facilement manipulable, et d'autre part de faciliter le décodage progressif. Nous n'allons pas rentrer dans les détails de cette transformée, mais nous aborderons néanmoins les points essentiels.

La compression par « transformée en ondelettes discrètes » consiste à décomposer une image par projections successives sur deux sous-espaces. L'un donnant l'allure générale de l'image (il s'agit de l'image en résolution moitié) et l'autre les détails. Si l'on interprète la transformée en ondelette discrète en terme de contraste et de fréquence, on peut dire que celle-ci consiste à extraire et garder telles quelles les zones de haute fréquences de l'image (le plus souvent les contours des objets) et de laisser la possibilité de soumettre les zones de basses fréquences à une deuxième compression. La décomposition que nous allons exposer par la suite est une version simplifiée de celle utilisée lors de la compression par Jpeg2000.

La décomposition se fait comme suit ; [WebOnd]

- On considère que E_p représente l'ensemble des images de taille 2^p x 2^p. (figure 5.5)
- On peut décomposer E_p de la manière suivante : $E_p = E_{p-1} + F_{p-1}$ ⟶ Détails
  ⟶ Allure générale
- On peut par la suite re-décomposer E_{p-1} de la même manière.

Cette décomposition peut se visualiser de la manière suivante :

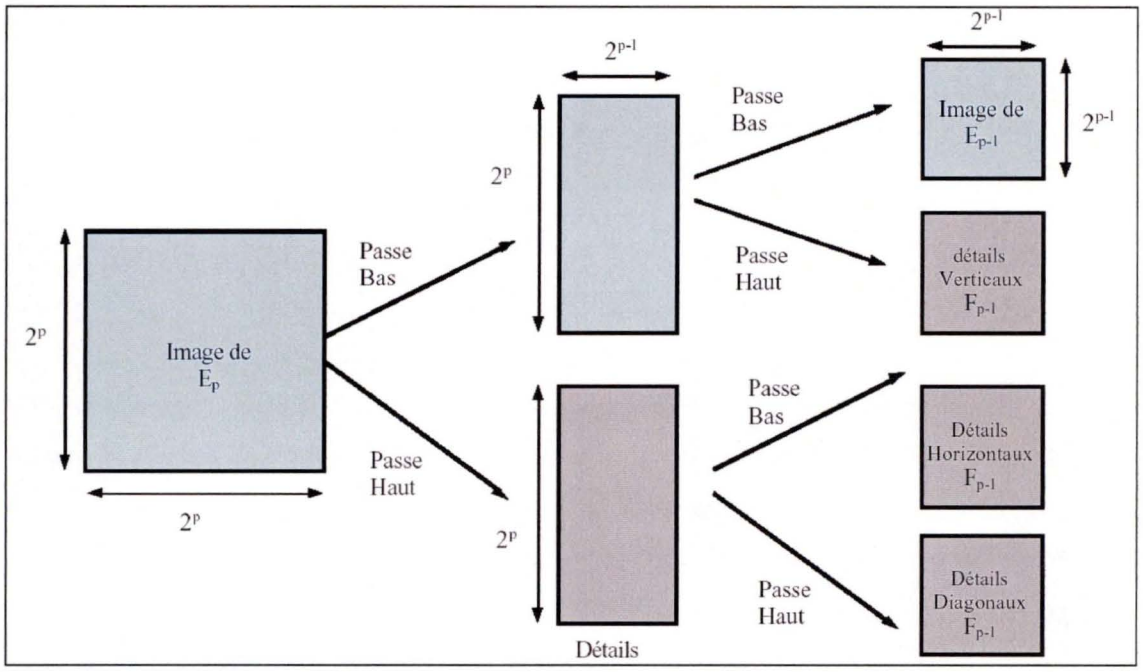


figure 5.5 : Schéma de la décomposition

Sur ce schéma nous voyons que les termes « Passe Haut » et « Passe Bas » sont utilisés. Il s'agit en fait de la décomposition de l'image suivant des hautes et basses fréquences, nous n'entrerons pas plus dans les détails. Nous nous contenterons simplement de montrer à travers la figure suivante un exemple de résultat obtenu après une transformée en ondelettes discrète.

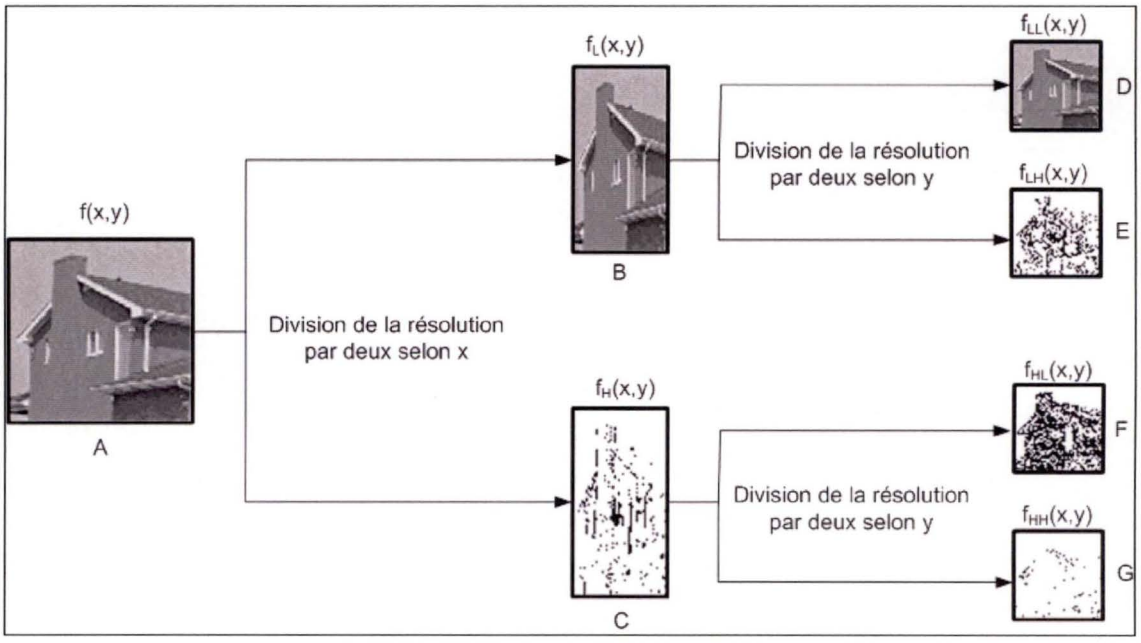


figure 5.6 : Transformée en ondelettes discrètes

Le résultat de cette transformation est une image d'approximation, à savoir l'image D de la figure ci-dessus. La résolution de cette image est divisée par deux par rapport à l'image d'origine. Les trois images de détails E, F, G nous donnent les erreurs entre l'image originale et l'image d'approximation D. Pour pouvoir reformer l'originale, il faut d'une part avoir l'image d'approximation D, et d'autre part avoir les trois images reprenant les erreurs entre l'image originale et l'image d'approximation. Cette transformation peut être répétée autant de fois que nécessaire afin d'obtenir le nombre voulu de sous bandes. La figure 5.7 montre une transformation avec 2 niveaux de décomposition.

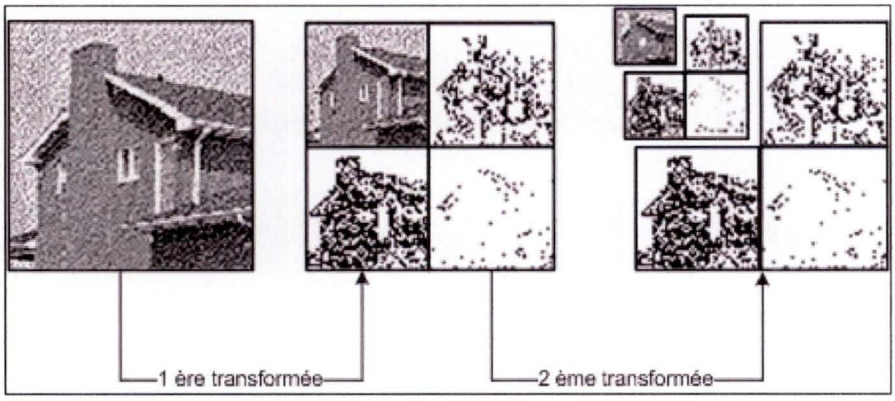


figure 5.7 Transformation avec 2 niveaux de décomposition

Lors de la transformation en ondelettes discrètes, différents filtres peuvent être utilisés pour la décomposition de l'image. Nous ne ferons ici que les énumérer. Pour plus d'information, mieux vaut se référer à la norme. Si la compression peut se faire avec perte, le filtre utilisant la paire (9,7) de Daubechies pourra être utilisé, si par contre la compression doit se faire sans perte de qualité, on pourra employer le filtre utilisant la paire (5,3) de Gall. Plus d'informations sur la décomposition en ondelettes sont disponibles en annexe.

- **La phase de quantification**

La phase de quantification consiste à approximer la valeur des coefficients d'ondelettes obtenus à l'étape précédente. Cette phase n'est utilisée que lors du codage avec perte, car lors du codage sans perte le cœfficient de quantification est laissé à 1. Cette étape permet donc de réduire la quantité d'information en ne conservant qu'un ordre de grandeur plus ou moins précis des cœfficients. La figure 5.8 illustre la fonction de quantification

utilisée dans Jpeg2000. Sur la figure, w est le cœfficient d'ondelettes à quantifier et $Q(w)$ l'index de quantification obtenu. [San01]

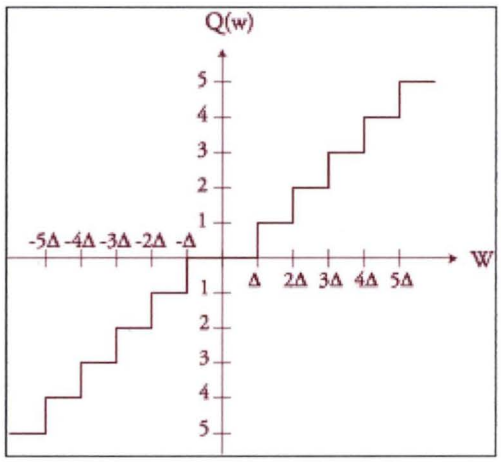


figure 5.8 : fonction de quantification dans Jpeg2000 [San01]

- **Le codage entropique**

Après les phases de transformation en ondelettes discrètes et de quantification, l'image originale représentée par la figure 5.9 a été divisée en sous bandes représentant des niveaux de résolutions différents comme indiqué sur la figure 5.10. Cette figure 5.10 représente les quatre niveaux de résolution découlant de trois décompositions effectuées lors de la phase de transformation en ondelettes discrètes. Sur la figure 5.10, chaque couleur représente une sous-bande, et chaque sous bande représente un niveau de résolution.

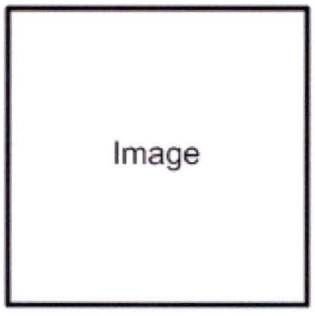


figure 5.9 : Représentation d'une image

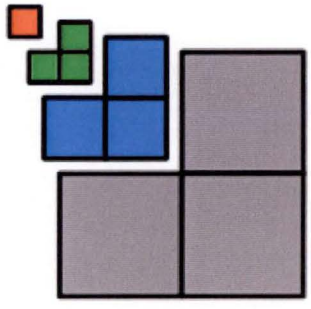


figure 5.10 Découpe en sous-bandes

Avant la phase de compression à proprement parler, l'image va encore être divisée, cette fois ci en blocs rectangulaires disjoints appelé *precincts*[15]. Un exemple de découpe en *precincts* est illustré par la figure 5.11. Un ensemble constitué de trois *precincts* représentant la même zone spatiale de l'image principale mais appartenant chacun à une sous-bande différente d'un même niveau de résolution est appelé paquet, un exemple de paquets est représenté par la figure 5.12

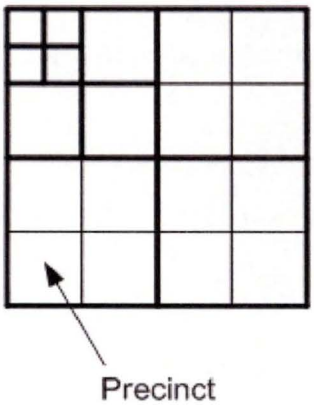


figure 5.11
Découpe en precincts

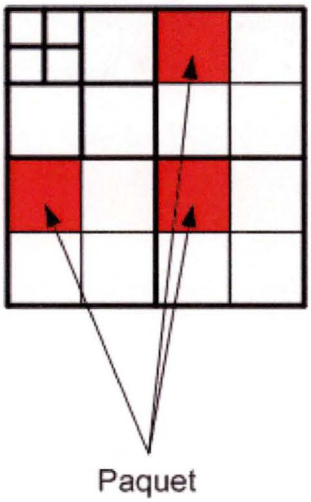


figure 5.12
Découpe en paquets

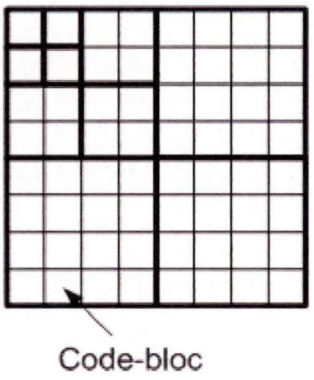


figure 5.13
Découpe en code-bloc

Une dernière division est effectuée avant le codage, il s'agit une fois de plus d'une division en rectangles disjoints appelé code-blocs. La figure 5.13 représente un exemple de division. Ces code-blocs constituent l'input du codeur arithmétique adaptatif avec contexte. Nous n'allons pas rentrer dans les détails des trois phases réalisées lors du codage entropique réalisées pour la compression, de plus amples informations sont disponibles en annexe.

- **L'allocation de débit**

La phase d'allocation de débit à pour but de fabriquer des *codestream* conformes à la norme. Cette phase peut varier fortement d'un codeur à l'autre. Une explication plus détaillée du contenu du *codestream* se trouve dans le point suivant.

---

[15] Traduction anglaise de « pourtour », « limite ».

5.6 Les informations du codestream

Une fois les différentes phases du codage effectuées, il nous reste un flux de données représentant l'image compressée, ce flux de données est appelé *codestream*. La constitution de ce *codestream* pour des fichiers de type Jpeg2000 est complexe étant donné que l'agencement du *codestream* permettra ou non l'utilisation des différentes fonctionnalités propre à Jpeg2000 à savoir le décodage progressif ou encore l'utilisation des régions d'intérêts. Nous avons vu précédemment comment se faisait la décomposition des images et nous avons énuméré les différentes subdivisions.

Nous allons maintenant passer en revue les différentes parties du *codestream* et nous allons montrer comment elles s'agencent.[16]

Le format des fichiers Jpeg2000 est basé sur un concept de boîte, ou chaque boîte est une séquence continue de données contenant des informations relatives à la longueur ou au type de données que cette boîte contient. Il existe aussi des « superboîtes » qui peuvent en encapsuler d'autres, ce qui donne une structure hiérarchique. La figure 5.14 représente les différentes parties constituant le flux de données d'un fichier compressé avec l'algorithme Jpeg2000. La superboîte principale représente l'image compressée sous forme de flux de données. Cette superboîte principale est constituée d'un ensemble de boîtes appelées paquets. Et chaque paquet est lui-même composé de code-blocs.

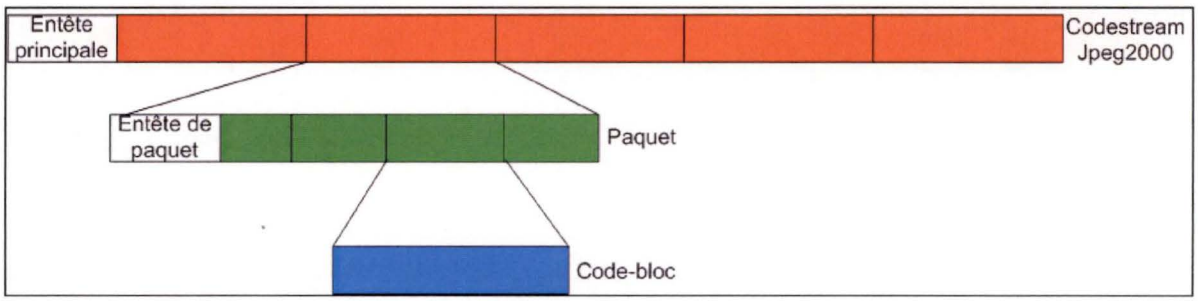


figure 5.14 : Constitution d'un *codestream* Jpeg2000 sans *layer*

[16] Une explication plus détaillée du *codestream* se trouve dans l'annexe 3.

La figure 5.14, représente une structure simplifiée. En effet si on veut profiter des multiples fonctionnalités proposées par Jpeg2000, comme le décodage progressif, il faut rajouter une boîte appelée *layer*[17] qui s'insert entre la superboîte contenant le flux de données et les paquets. Un exemple de cette structure hiérarchique est présenté par la figure 5.15. Cette nouvelle boîte appelée *layer* contiendra un ensemble de paquets. La décompression d'une image Jpeg2000 se fait le plus souvent de façon séquentielle. On peut donc dire que l'agencement des *layers* ainsi que leur contenu définiront la manière dont l'image sera décodée. Il faut savoir que chaque *layer* contient une partie des informations d'une certaine zone de l'image originale. Donc si on veut coder une image en utilisant la fonctionnalité du codage par région d'intérêt, il faut que les paquets représentant la zone intéressante soient décodés avant les autres. La figure 5.16 montre un exemple de décomposition en *layer* pouvant être à l'origine de la figure 5.17. Il est cependant évident que la figure 5.17 n'est pas constituée que de quatre code-blocs. Cet exemple est là à titre indicatif.

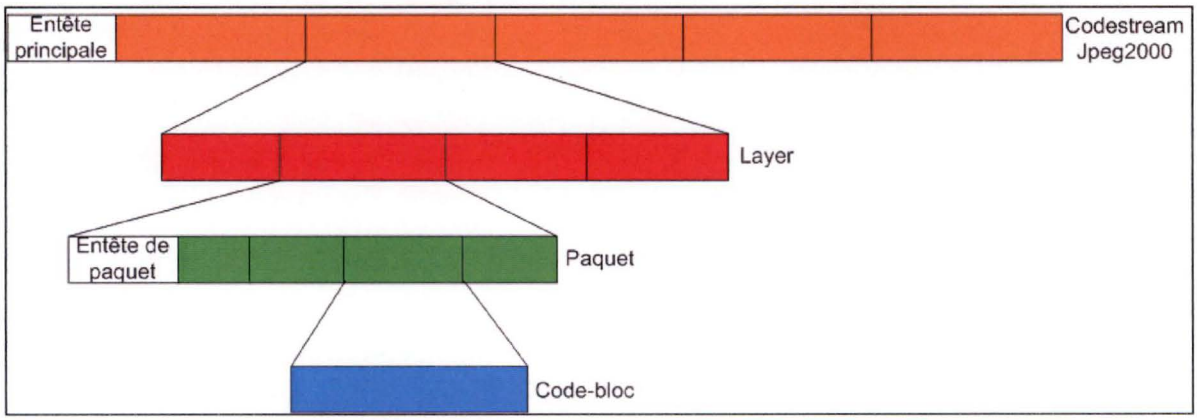


figure 5.15 Constitution d'un *codestream* Jpeg2000 avec *layer*

[17] Traduction anglaise de « couche ».

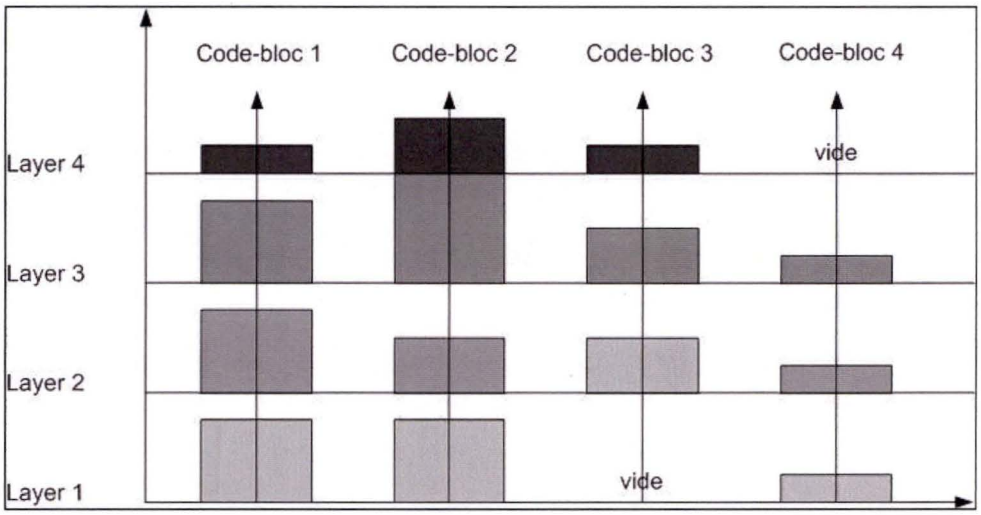


figure 5.16 Composition des *layers* en fonction des code-blocs

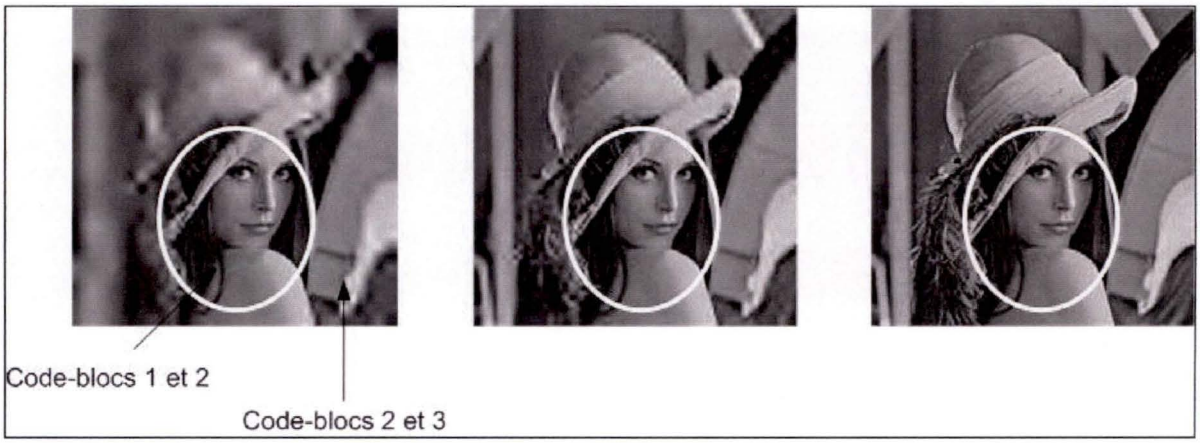


figure 5.17 : Exemple d'image codée par région d'intérêt

Pour la réalisation des applications exposées dans la partie 3 Analyse et résultats, nous n'avons pas utilisé le concept de *layer*. Nous nous sommes simplement servis d'une structure de données identique à celle représentée par la figure 5.18. Par la suite, la superboîte principale sera appelée *codestream* et un paquet sera considéré comme étant un *tile*. Toutes les images que nous utiliserons dans la partie 3 seront donc décomposables en *tiles*.

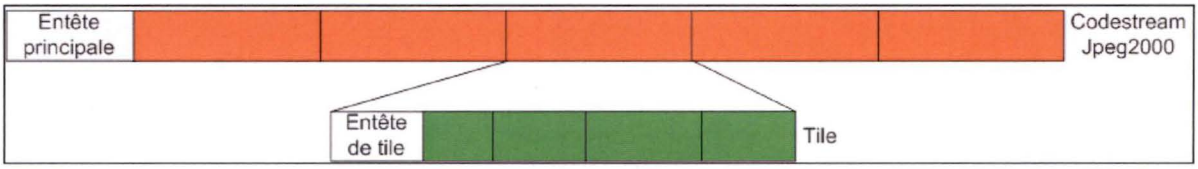


figure 5.18 : *Codestream* Jpeg2000 semblable à celle utilisée dans la partie 3

La norme décrivant le format des fichiers Jpeg2000 définit un certain nombre de marqueurs[18] servant à l'élaboration du *codestream*. Le tableau 5.1 reprend ces principaux marqueurs qui seront utilisés par la suite, soit dans un main header ou dans un tile header.

	Nom	Code hexadécimal
Marqueurs de délimitation		
Début du codestream	SOC	0xFF40
Début d'un Tile	SOT	0xFF90
Début des données	SOD	0xFF93
Fin du codestream	EOC	0xFFD9
Marqueurs d'informations fixes		
Taille de l'image et des tiles	SIZ	0xFF51

Tableau 5.1 : Tableau regroupant les principaux marqueurs

La figure suivante présente les différent marqueur présent dans l'entête principale, l'entête des tiles ainsi que celui utilisé pour indiquer la fin du *codestream*.

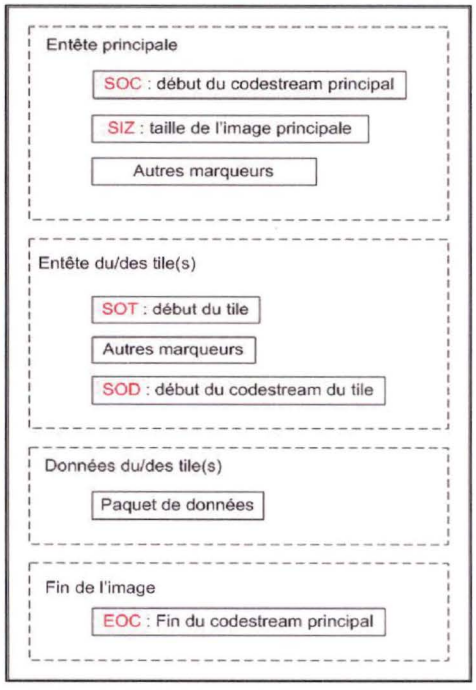


figure : 5.19 : Marqueurs présent dans les entêtes.

[18] Traduit en anglais par « tag » ou « marker ».

5.7 Comparaison de Jpeg2000 et Jpeg

Comparer les performances des méthodes de compression n'est pas chose facile car plusieurs facteurs entrent en ligne de compte. Ces différents facteurs sont entre autre le taux de compression, la qualité de l'image, le temps de codage et les ressources nécessaires. Nous allons tout d'abord nous intéresser aux fonctionnalités que nous offre Jpeg2000 par rapport aux méthodes de compression conventionnelles. La figure 5.20 montre les choix possible lors de l'encodage ainsi que lors du décodage d'une image avec une méthode de compression traditionnelle comme Jpeg. On peut voir qu'il n'y a pas de choix laissé à l'utilisateur lors du décodage.

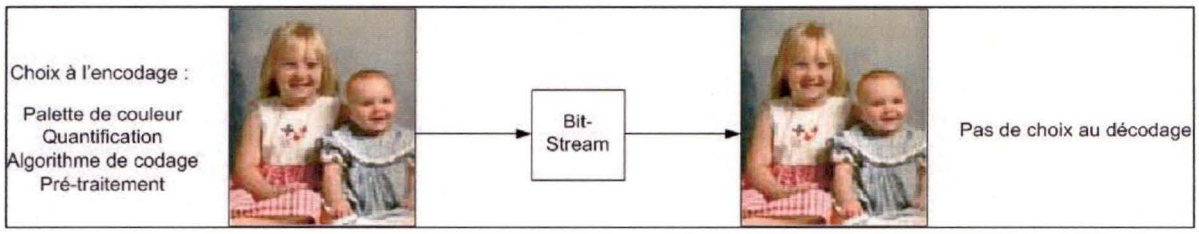


figure 5.20 : Image codée et décodée avec la méthode Jpeg

La méthode de compression Jpeg2000 permet quant à elle un nombre de choix plus important lors de l'encodage ainsi que la possibilité de décoder l'image de différentes manières selon l'usage que l'on veut en faire.

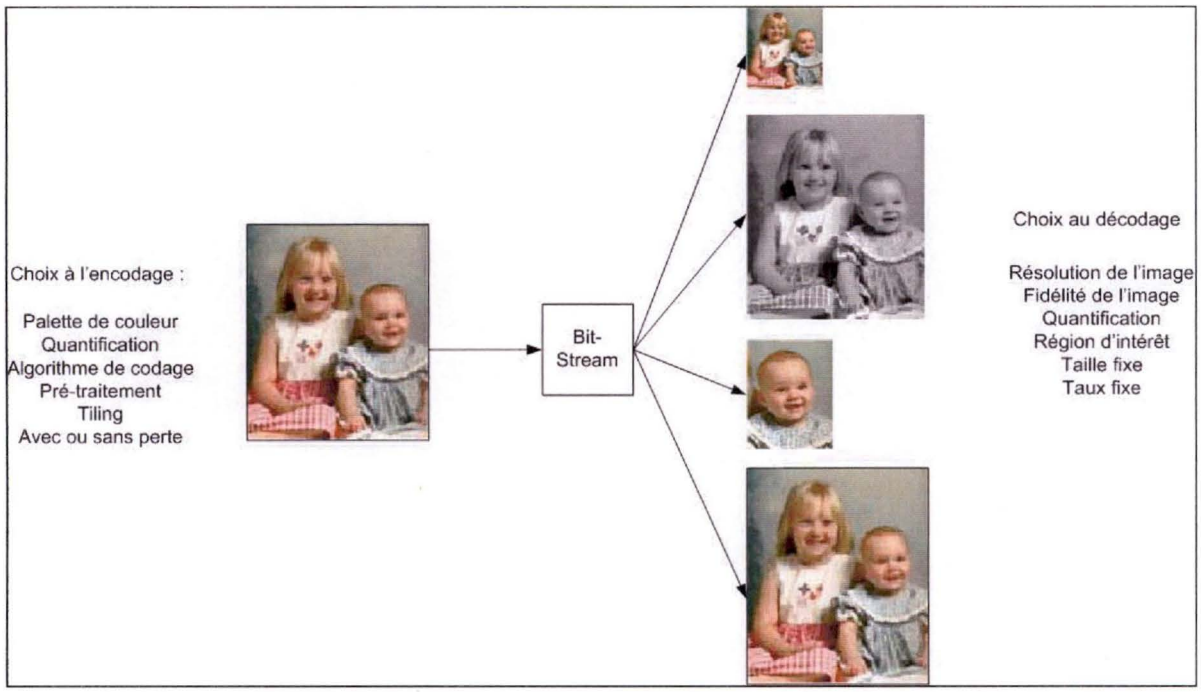


figure 5.21 : Image codée et décodée avec la méthode Jpeg2000

Un avantage de la méthode de compression réside dans ses nombreuses fonctionnalités énumérées au point 5.4

Au niveau de la qualité d'image, la compression avec Jpeg2000 donne des meilleurs résultats aussi bien à des forts qu'à des faibles taux de compression. Néanmoins, la différence avec Jpeg est plus importante à des taux de compression élevés, comme nous pouvons le constater sur les figures 5.22 et 5.23. Ces différences sont notamment dues à la méthode de compression. Celle de Jpeg2000 se fait par ondelettes, au contraire de celle de Jpeg qui se fait par blocs de 8 pixels sur 8 comme dans la méthode DCT[19] (Jpeg). Les blocs Jpeg de 8 x 8 sont quantifiés indépendamment les uns des autres ce qui ne permet pas de réduire les redondances au delà d'un bloc. Au contraire de la compression par ondelettes qui est une méthode globale sur toute l'image. Cet avantage se traduit par une efficacité encore plus importante sur les grosses images.

[19] Discrete Cosine Transform

figure 5.22 :
Image codée avec Jpeg à 0,125bpp

figure 5.23 :
Image codée Jpeg2000 à 0,125bpp

figure 5.24 :
Image codée avec Jpeg à 0,5bpp

figure 5.25 :
Image codée Jpeg2000 à 0,5bpp

Au niveau du temps de compression, l'algorithme de compression Jpeg est beaucoup plus rapide que celui de Jpeg2000, aussi bien au niveau du temps de codage qu'au niveau du temps de décodage.

5.8 Le logiciel jj2000

jj2000 est un logiciel programmé en java qui implémente l'algorithme de compression Jpeg2000 tel qu'il est décrit dans la première partie de la norme « ISO/IEC 15444-1 ». Le logiciel jj2000 est l'un des seuls qui implémente l'algorithme de codage relatif à la technique de compression Jpeg2000 et qui soit gratuit et open-source. Le logiciel jj2000 a été développé par Canon, l'institut fédéral Suisse et le département recherche de l'entreprise Ericsson.

L'algorithme de codage du logiciel jj2000 est semblable à celui présenté dans la section 5.5. Cet algorithme sera utilisé dans la partie 3 Analyse et Résultats. Lors des différentes phases de la compression, plusieurs facteurs peuvent être fixés par l'utilisateur, notamment le nombre de niveaux de décomposition lors de la phase de décomposition en bandelettes discrètes ou encore le fait que la compression se fasse avec ou sans perte. Nous utiliserons principalement les paramètres disponibles lors de la phase de découpe en *tiles*. En effet le module de gestion des *tiles* de jj2000 nous donne la possibilité de compresser une image qui pourra ensuite être insérée dans une autre image plus grande. Il suffit pour cela de connaître la position que l'image à insérer aura dans l'image principale.

Par exemple si nous disposons d'une part une image de 300x400 pixels compressée avec Jpeg2000 et constituée de douze tiles d'une dimension de 100x100 pixels, une telle image est représentée par la figure 5.26. D'autre part nous disposons une image de 100x100pixels. Grâce au paramètre proposé par jj2000, il est possible de compresser la deuxième image de sorte qu'on puisse par la suite l'insérer dans la première. Il faut pour cela fixer le paramètre « ref » présent dans jj2000 à la valeur voulue. Dans notre exemple, nous le fixerons à la valeur (200,200) ce qui nous permettra par la suite de placer la 2ème image dans la première à l'endroit indiqué sur la figure 5.26. Nous verrons ceci plus en détail dans le chapitre 9 de la troisième partie : analyse et résultats

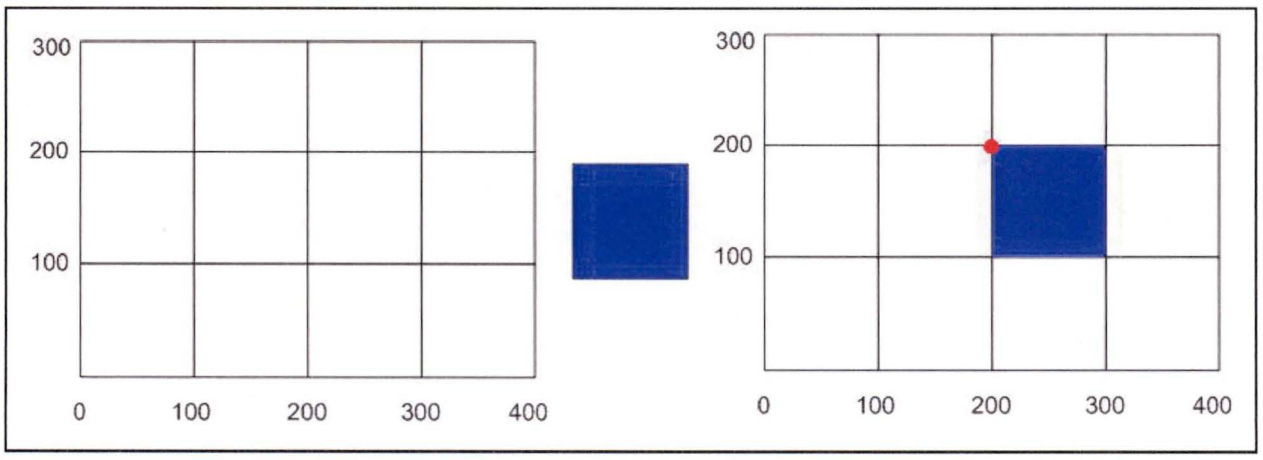


figure 5.26

Partie 3 : Analyse et résultats

Chapitre 6 : Rappel des objectifs et choix principaux

L'objectif principal du travail que nous avons eu à réaliser fut de contribuer à l'élaboration, déjà en cours, d'un instrument de laboratoire permettant l'observation virtuelle d'un prélèvement biologique. Les travaux déjà accomplis à l'hôpital universitaire de Mont-Godinne par les étudiants précédents nous servirent de base. Notre travail a principalement consisté à développer le programme réalisé par Louis Zuyderhoff dans [Zuy03]. Les phases principales de ce programme ont été exposées dans le chapitre XX de la partie deux.

Nous allons commencer par analyser les limites de la solution proposée dans [Zuy03] pour ensuite montrer comment nous avons tenté d'y apporter des améliorations. La première et la plus importante limite est celle de la taille des frottis à numériser. Comme nous l'avons vu dans le chapitre 4 de la partie 2, la solution proposée dans [Zuy03] est limitée par la taille de la mémoire centrale, et ce à cause de l'algorithme de coregistration qui nécessite le chargement en mémoire de toutes les images servant à la construction d'une méga-image. Le chapitre 7 portera donc sur la présentation d'une tentative de solution à la limitation imposée par l'utilisation de la mémoire centrale. Cette solution sera établie sur base d'une coregistration des bords des méga-images ainsi que sur une plus grande utilisation du disque dur.

D'autres problèmes ont été soulevés dans la conclusion de [Zuy03] mais ils étaient dûs à des problèmes matériels inhérent au microscope électronique et à l'autofocus.

Si l'on se réfère au tableau présenté au chapitre 2 reprenant la taille en GB du volume nécessaire pour stocker une lame de 20x25 mm à un grossissement 100x, on constate que ce volume s'élevait à environ 218 GB. On voit donc bien l'impossibilité d'utiliser un algorithme de coregistration en mémoire Ram pour numériser une lame complète. Même si l'on réduit la zone à analyser à $1cm^2$ ce qui correspond aux yeux du biologiste Yvan Cornet, cytologiste spécialiste en hématologie à Mont Godinne, le minimum acceptable pour une analyse correcte, le volume des images à manipuler reste énorme, à savoir environ 43.6 GB. On peut donc conclure que la coregistration des images ne peut se faire exclusivement en mémoire RAM.

L'objectif principal de ce mémoire est de présenter une solution permettant la numérisation d'une surface suffisamment grande d'une lame. C'est-à-dire réaliser un frottis numérique comprenant suffisamment d'informations que pour permettre aux cytologistes d'émettre un avis.

Le choix de java comme langage de programmation pour l'élaboration de l'application réalisée fut dicté d'une part par le fait que l'application déjà présente était programmée en java, et d'autre part par la réutilisabilité et la facilité qu'offre java pour la découpe en modules. Le choix de java fut aussi fait en fonction de mon cursus.

De par ses nombreuses fonctionnalités, la norme de compression Jpeg2000 s'est révélée un choix adéquat. En effet, comme nous avons pu nous en rendre compte dans le chapitre 5 de la partie deux, Jpeg2000 offre de nombreuses facilités pour la manipulation des grandes images. L'utilisation de l'algorithme jj2000 fut quant à elle obligatoire car, comme cela a été expliqué dans la partie deux, jj2000 constitue la seule implémentation en java gratuite et open source.

Chapitre 7 : La création des giga-images

7.1 Présentation de la solution

Comme nous l'avons déjà souligné, il est matériellement impossible de charger plusieurs méga-images parallèlement en Ram. Il a donc fallu trouver un moyen de coregistrer ces méga-images sans les charger entièrement en mémoire. La solution pour laquelle nous avons opté consiste à découper et sauvegarder les bords contigus des méga-images et d'effectuer ensuite la coregistration. Ceci solutionne le problème car il n'y a qu'une méga-image en Ram à la fois, et la coregistration se fait sur une partie ce celle-ci, à savoir les bords.

Pour rappel, nous avons défini une méga-image comme étant la plus grosse image qu'il est possible de construire en mémoire Ram. Le concept de giga-image est quant à lui associé à une image constituée d'un ensemble de méga-images et dont la coregistration ne se fait pas uniquement en Ram.

La solution que nous avons développée s'articule autour d'une matrice (3,3) regroupant tous les cas de figure pouvant être rencontrés lors de la coregistration de méga-images. Cette matrice est illustrée par la figure 7.1. Chaque carré représente une méga-image. Les bords verts sont les bords qui seront découpés et sauvegardés sur le disque dur et les bords rouges, quant à eux, seront découpés et coregistrés avec les bords verts correspondants. Nous pouvons constater que cette matrice ne se limite pas à 9 méga-images mais peut être agrandie aussi bien verticalement que horizontalement. Les neufs cas de figure présentés ci-dessous sont exhaustifs. Comme cette matrice sert pour l'assemblage des méga-images nous l'appellerons matrice « A ».

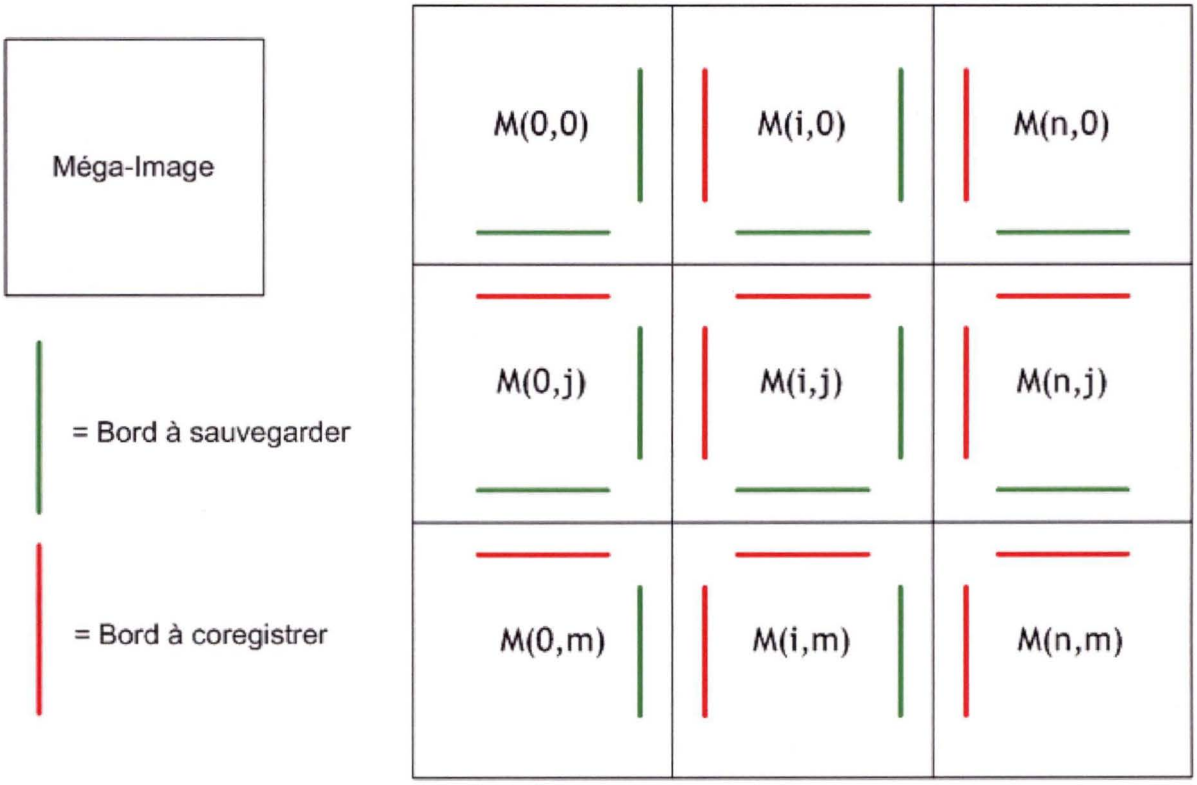


figure 7.1 : matrice A

Pour que cette solution soit fonctionnelle il faut que l'ordre de fabrication des méga-images soit respecté et se fasse ligne par ligne, de gauche à droite. En effet, si nous voulons par exemple coregistrer la méga-image (i,0) avec la méga-image (i-1,0), il faut que le bord droit de la méga-image (i-1,0) ait été créé et que son bord droit ait été sauvegardé sur le disque dur avant que ne soit créée l'image (i,0).

La figure 7.2 illustre l'enchaînement des différentes phases rentrant en ligne de compte lors de la fabrication d'une giga-image, tandis que la figure 7.3 reprend les différentes sous-phases qui font partie de la phase « traitement des méga-images ».

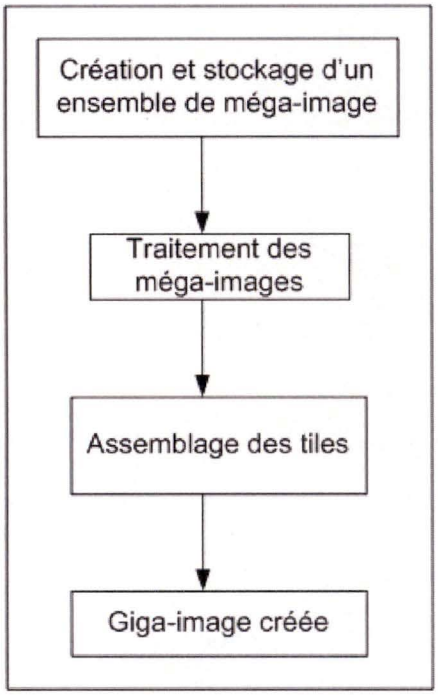


figure 7.2 : Phases de création d'une giga-image

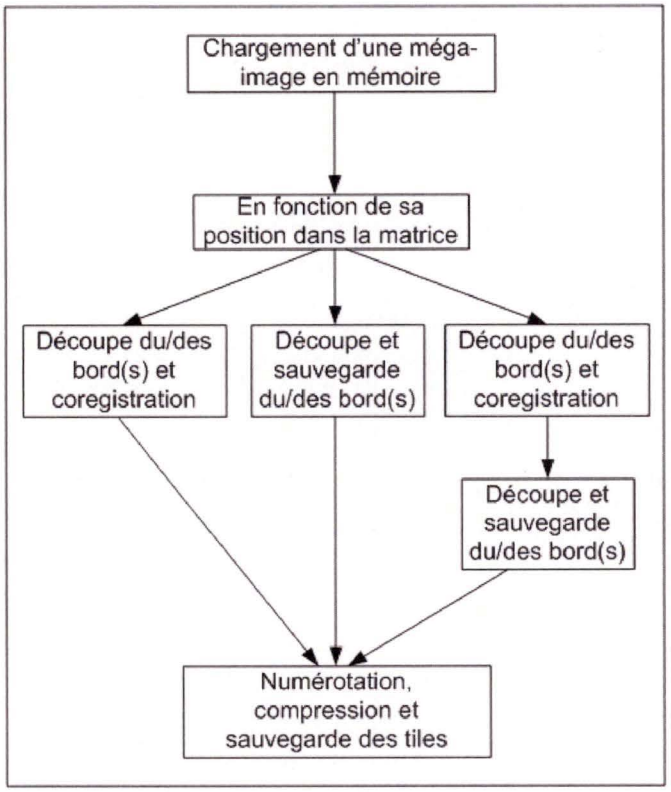


figure 7.3 : Sous-phases intervenant lors de la phase de traitement

Nous allons voir dans le point suivant un exemple de construction d'une giga-image issue de deux méga-images coregistrées horizontalement. Nous détaillerons ensuite dans les chapitres suivants certaines phases de ce processus.

7.2 Exemple de construction d'une giga-image

L'exemple qui suit présente l'enchaînement des différentes phases faisant partie de la construction d'une giga-image. Cette giga-image sera constituée de deux méga-images coregistrées horizontalement.

La première étape consiste à charger en mémoire la première méga-image. Pour plus de facilité nous l'appellerons « M(0,0) » (figure 7.4), pour méga-image ayant la position (0,0). La seconde image sera nommée « M(1,0) » (figure 7.7), pour méga-image ayant la position (0,1). Les positions de ces méga-images sont importantes car ce sont elles qui décident du traitement que ces méga-images vont subir.

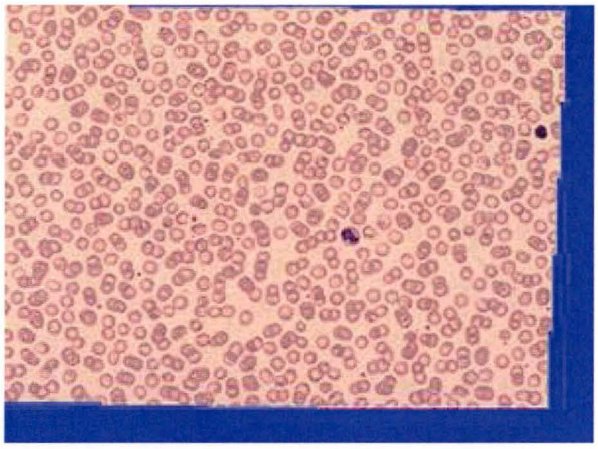


figure 7.4 : Méga-image « M(0,0) »

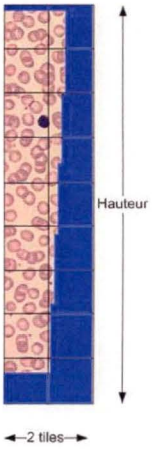


figure 7.5 : Bord droit de « M(0,0) »

Une fois chargée en mémoire, « M(0,0) » va être traitée en fonction de sa position dans la matrice de la figure 7.1. Nous voyons par rapport à cette matrice que, pour la méga-image de position (0,0), la phase de traitement consiste à découper et sauvegarder le bord droit et le bord inférieur de celle-ci. La taille des bords est choisie en fonction de la taille des *tiles* composant l'image. Pour notre exemple nous avons pris des bords d'une longueur de deux *tiles* et d'une hauteur correspondant à la hauteur de l'image. Le bord droit de « M(0,0) » est représenté par la figure 7.5. Une fois le bord découpé, celui-ci est sauvegardé sur le disque dur. On fait la même chose pour le bord inférieur.

La phase suivante est la phase de compression : chaque *tile* constituant l'image « M(0,0) » est compressé, numéroté et sauvegardé. Cette phase de compression est expliquée plus en détail dans le chapitre quatre.

Quand la phase de traitement de la première méga-image est finie, on charge en mémoire la méga-image suivante, dans notre cas « M(1,0) ». On la traite ensuite en fonction de sa position dans la matrice. « M(1,0) » est considérée comme étant la dernière image de la ligne constituant la giga-image car nous construisons une giga-image de deux méga-images. Il faut donc découper et coregistrer le bord gauche et découper et sauvegarder le bord inférieur. La figure 7.7 montre « M(1,0) » et la figure 7.6 montre le bord gauche d'une longueur de deux *tiles* et d'une hauteur correspondant à la hauteur de « M(1,0) ».

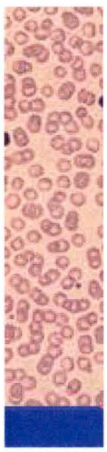


figure 7.6 : Bord gauche de « M(1,0) »

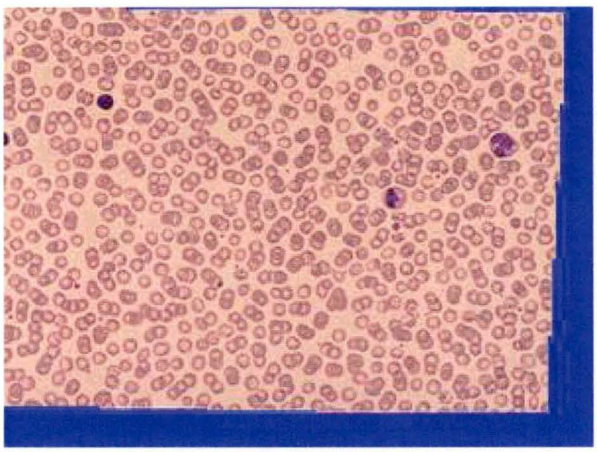


figure 7.7 : Méga-image « M(1,0) »

La phase suivante consiste à coregistrer le bord droit de « M(0,0) » (figure 7.8) avec le bord gauche de « M(1,0) » (figure 7.9). Pour effectuer cette coregistration, les deux bords doivent être chargés en mémoire. Le but de cette coregistration est de connaître l'endroit ou commence la méga-image « M(1,0) » dans la méga-image « M(0,0) ». Suite à cela nous pouvons compléter le bord gauche de « M(0,0) » avec le bord droit de « M(1,0) ». Cette phase de coregistration est détaillée dans le chapitre 8. Une fois la coregistration effectuée nous obtenons une image (figure 7.10) de la même dimension que le bord gauche de « M(0,0) » mais contenant non seulement les informations du bord gauche mais aussi les informations du bord droit qui ont été nécessaires pour compléter le bord gauche.

figure 7.8

figure 7.9

figure 7.10

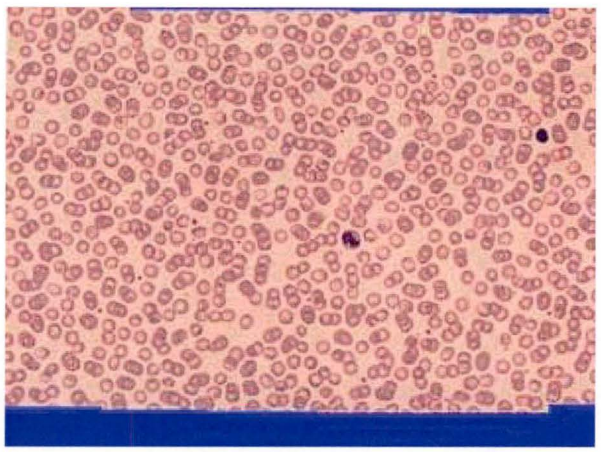


figure 7.11 : Partie gauche de la giga-image

Suite à cette coregistration, l'image « M(1,0) » doit encore subir quelques transformations avant de pouvoir être compressée. Ces transformations sont des décalages horizontaux ou verticaux pour ajuster l'image car suite à la coregistration, le bord gauche de celle-ci a été modifié. Une fois ces transformations effectuées tous les *tiles* qui constituent l'image « M(1,0) » sont numérotés, compressés et sauvegardés. La figure 7.11 montre comment sera l'image « M(0,0) » une fois que son bord gauche aura été complété.

La dernière phase consiste à assembler tous les tiles qui ont été précédemment compressés et sauvegardés afin d'obtenir la giga-image résultant des deux méga-images, ce qui donne la figure 7.12.

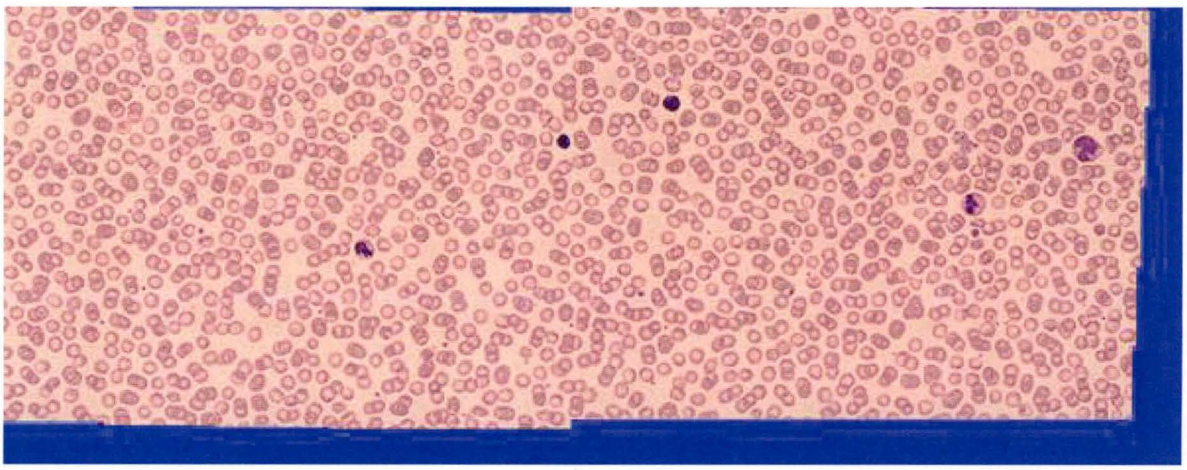


figure 7.12 : Giga-image contenant M(0,0) et M(1,0)

Nous avons vu au travers de cet exemple les différentes phases que sont le chargement d'une méga-image en mémoire, la découpe du ou des bords, la coregistration de deux bords, la numérotation et compression des différents *tiles* et l'assemblage des *tiles* Ces trois dernières phases vont faire l'objet d'une étude plus approfondie dans les deux chapitres qui suivent.

Chapitre 8 : La phase de Coregistration

Ce chapitre va, dans un premier temps, expliquer plus en détail la phase de coregistration qui intervient dans la construction des giga-images et va ensuite présenter les problèmes rencontrés lors de cette phase de coregistration. Comme nous l'avons vu dans l'exemple du chapitre précédent, la coregistration se fait entre deux images et plus précisément entre les bords de deux méga-images contiguës. La première image va être appelée « image maître » et la seconde « image esclave ».

Pour rappel, « on entend par coregistration le processus d'alignement ou de mise en correspondance de deux images, appelées « image maître » et « image esclave » recouvertes entièrement ou partiellement par une même zone. Cette zone sous-entend donc que les images représentent entièrement ou partiellement une même information que l'on appellera la « scène observée » ». [Zuy03]

Le processus de coregistration est un processus complexe, il est expliqué et utilisé ci-dessous de façon simplifiée. En effet, d'une part nous ne coregistrons que deux vecteurs entre eux, et non deux images entre elles, et d'autre part nous ne tenons pas compte de la possibilité d'avoir des décalages obliques entre deux méga-images.

La phase de coregistration se décompose en trois étapes ;

une étape de pré-traitement qui va préparer l'image maitre
une étape de coregistration et
une étape de post-traitement

Les trois étapes sont présentées ci après :

8.1 L'étape de pré-traitement

Avant de passer à l'étape de coregistration à proprement parler, l'image maître va subir quelques transformations. La première étape consiste à trouver les endroits sur l'image où commence le remplissage[20]. Et ce, aussi bien verticalement que horizontalement. Les flèches rouges que nous voyons sur la figure 8.1 indiquent les parties de l'image qui vont être découpées pour donner la figure 8.2. C'est à partir de cette image que l'étape de coregistration va débuter, elle sera donc appelée image maître.

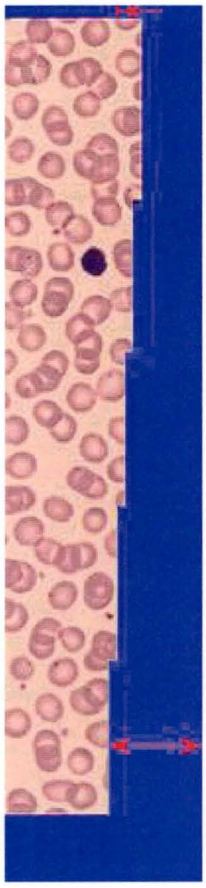


figure 8.1

figure 8.2

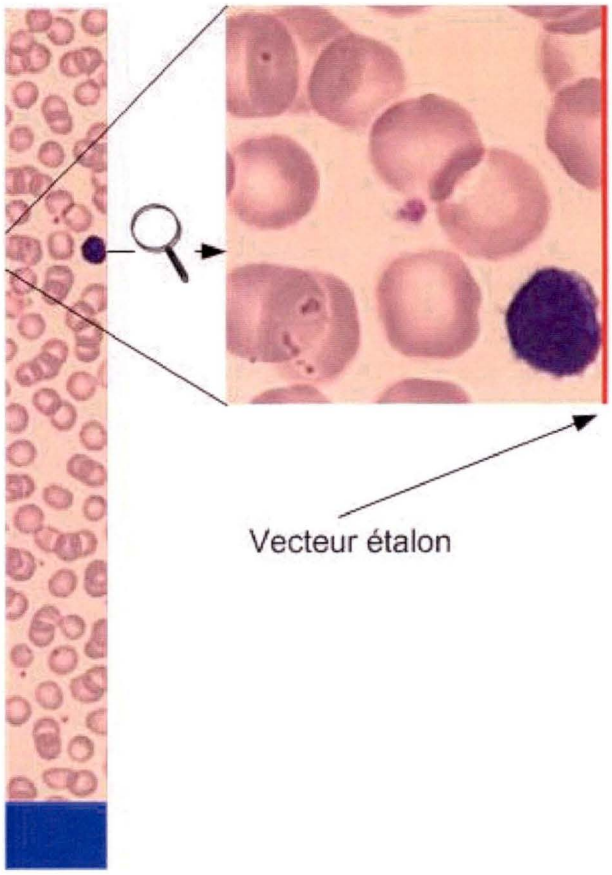


figure 8.3 : Choix du vecteur étalon

[20] Il s'agit en fait de la zone en bleu sur la figure 8.1.

8.2 L'étape de coregistration

La coregistration de deux images se fait en comparant l'image maître et l'image esclave. Une comparaison de toute l'image prendrait beaucoup trop de temps. C'est pour cela que dans la plupart des cas, seule une partie de l'image est comparée. La solution que nous avons mise en œuvre se divise en plusieurs phases.

La première tâche consiste à trouver dans l'image maître un vecteur[21], qui nous servira d'étalon. Ce vecteur est d'une longueur et d'une largeur fixes. Ces deux valeurs n'ont pas fait l'objet d'une étude approfondie par manque de temps. La seule contrainte concernant ce vecteur, c'est qu'il doit être choisi dans la zone qui se trouve dans l'image maître et dans l'image esclave, la ligne rouge présente dans la figure 8.3 ci-dessus correspond au vecteur étalon de la figure 8.2

La seconde tâche a pour but de retrouver, dans l'image esclave, le vecteur étalon précédemment choisi dans l'image maître. Pour retrouver ce vecteur, une zone imaginaire de forme rectangulaire située en haut à gauche de l'image esclave est définie. Cette zone définie sert de point de départ pour un ensemble de vecteurs qui vont tour à tour être comparés[22] avec le vecteur étalon. Nous pouvons voir sur la figure 8.4 que la zone rectangulaire sert de commencement à l'ensemble des vecteurs de couleur rouge, ceux-ci allant être comparés au vecteur étalon. Sur cette figure se trouve aussi un vecteur vert. C'est ce vecteur qui, suite à la comparaison, a correspondu le mieux au vecteur étalon de l'image maître présenté sur la figure 8.2.

Grâce à la seconde tâche, nous savons maintenant que la partie de l'image esclave située à droite du vecteur vert correspond au prolongement de l'image maître. Il nous faut donc compléter l'image maître avec la partie de l'image esclave située à droite du vecteur vert. La partie située à droite de l'image esclave est représentée par la figure 8.5. Cette partie

---

[21] Tableau à une seule dimension, dans notre cas une ligne de pixel.
[22] On compare les pixels des vecteurs deux à deux.

peut ensuite être assemblée avec l'image 8.6 représentant l'image maître, ce qui donne la figure 8.7.

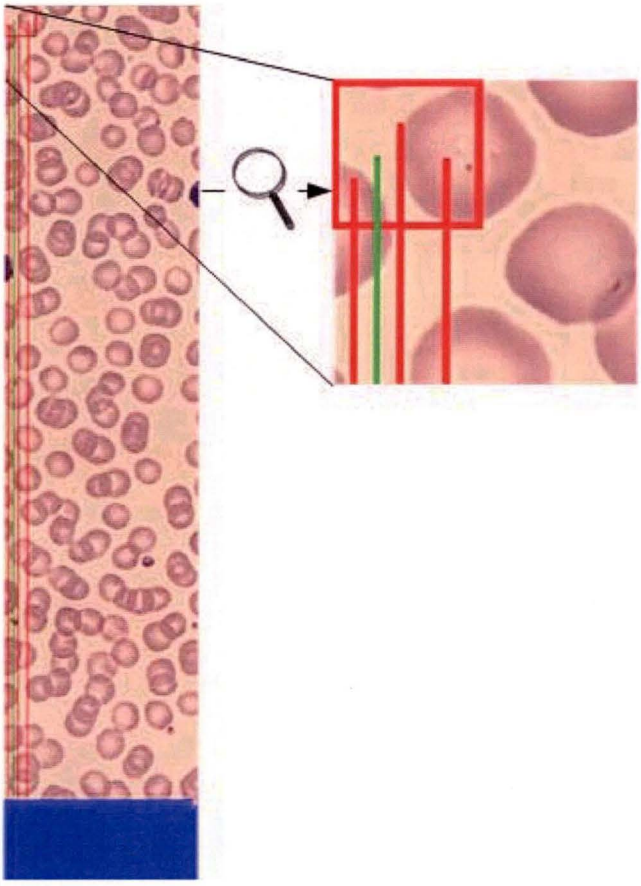


figure 8.4 : Zone de sélection

figure 8.5

figure 8.6

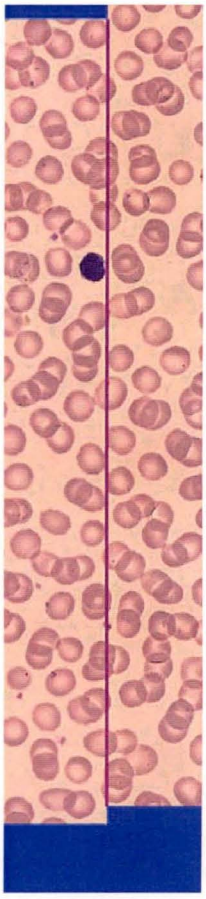


figure 8.7

8.3 L'étape de post-traitement

L'étape de post-traitement s'effectue sur l'image esclave. Dans l'étape précédente nous avons découpé une partie de l'image esclave pour compléter l'image maître. On considère que cette partie d'image a une dimension de d1 en longueur et d2 en largeur. Si nous voulons que les 2 images s'assemblent parfaitement il faut décaler l'image esclave de d2 pixels horizontalement.

Une fois ces trois étapes effectuées, nous obtenons une image maître dont le bord droit est rempli et une image esclave qui s'assemble parfaitement avec l'image maître. Ce qui nous permet de construire une giga-image illustrée par la figure 8.8

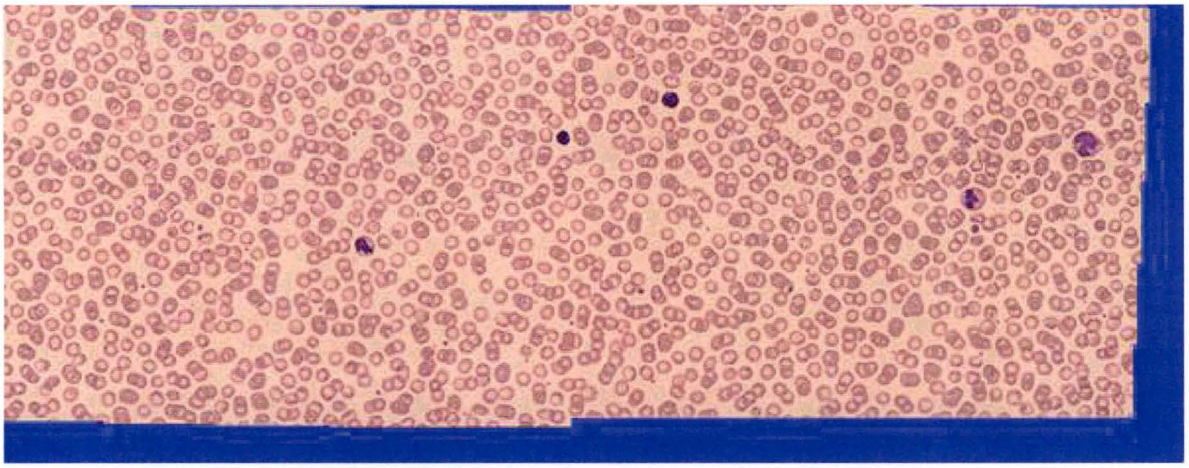


figure 8.8 : Giga-image

8.4 Les problèmes rencontrés

Plusieurs problèmes ont été rencontrés lors de la phase de coregistration, nous allons les détailler ci-dessous.

Le premier problème qui se pose apparaît lors de la comparaison de l'ensemble des vecteurs de l'image esclave avec le vecteur étalon de l'image maître. Cette comparaison s'effectue sur un seul vecteur ce qui n'est pas toujours suffisant, car il arrive parfois que deux méga-images contiguës aient des différences de luminance ou de contraste entre elles. Ces différences pouvant impliquer des erreurs d'alignement[23] lors de la coregistration.

Un autre problème se pose lorsque les petites images qui sont à la base des méga-images n'ont pas été bien coregistrées. Une mauvaise coregistration se traduit par des décalages entre les petites images constituant les méga-images. Ces décalages peuvent provenir de plusieurs origines différentes. On peut par exemple mettre en cause la précision

---

23

du microscope électronique, car comme nous l'avons déjà vu dans la première partie, un pixel numérisé au zoom 100x représente 0.08 µm (1 micron = 1 millionième de millimètre). On peut donc aisément comprendre que des décalages peuvent arriver. Une seconde origine des décalages peut provenir de la différence de netteté entre les images. Cette différence de netteté tient son origine du fait que le support sur lequel repose les lames est légèrement incliné, Cette légère inclinaison a pour conséquence une différence de netteté entre 2 images prises à des endroits différents de la lame. Ces problèmes de netteté sont néanmoins atténués par la présence d'un autofocus matériel.

Certains problèmes ont pour cause un recouvrement trop ou trop peu importants de deux méga-images. Ces recouvrements sont difficiles à fixer car ils résultent d'une part des coordonnées choisies lors de l'acquisition des images du microscope, et d'autre part de l'algorithme de coregistration issu de [Zuy03]. Une étude plus approfondie de ces deux facteurs réduirait le nombre de problèmes de coregistrations des méga-images.

Les problèmes qui sont présentés ci-dessus ont pour conséquence des décalages de quelques pixels dans la giga-image au niveau de la jonction entre les deux méga-images.

L'endroit dans une giga-images ou nous avons constaté le plus de décalages se situe au niveau de la zone de recouvrement de quatre méga-images. Cette zone à une dimension de deux *tiles* de longueur et un tile de hauteur et est représentée par un rectangle rouge sur la figure 8.9. Elle est la plus sujette à des décalages car elle résulte de quatre phases de coregistration.

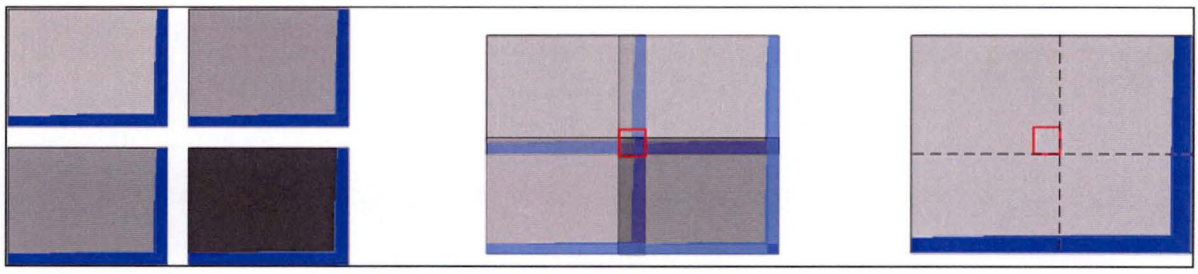


figure 8.9

Le stockage et la compression font l'objet du chapitre suivant.

Chapitre 9 : L'encodage avec Jpeg2000

La norme de compression Jpeg2000 est omniprésente dans la réalisation de cette application. Ce chapitre a pour but d'expliquer les différentes phases de la construction des giga-images qui font appel à Jpeg2000. Il y a d'une part la phase de « numérotation, compression et sauvegarde des *tiles* », et d'autre part la phase d' « assemblage des *tiles* ».

9.1 La phase de numérotation, compression et sauvegarde des *tiles*

L'une des particularités de la méthode de compression Jpeg2000 réside dans le fait qu'une image peut être décomposée en *tiles,* et que chacun de ces *tiles* peut être compressé indépendamment des autres. Nous nous sommes servis de cette fonctionnalité lors de la phase de « numérotation, compression et sauvegarde des tiles ». Comme nous pouvons voir sur la figure 9.1, une méga-image peut être décomposée en un ensemble de *tiles*. Chaque tile est numéroté en fonction de sa place dans la méga-image. Chacun des tiles a une dimension de 100 x 100 pixels. La méga-image a donc une dimension de 1200 x 900 pixels.

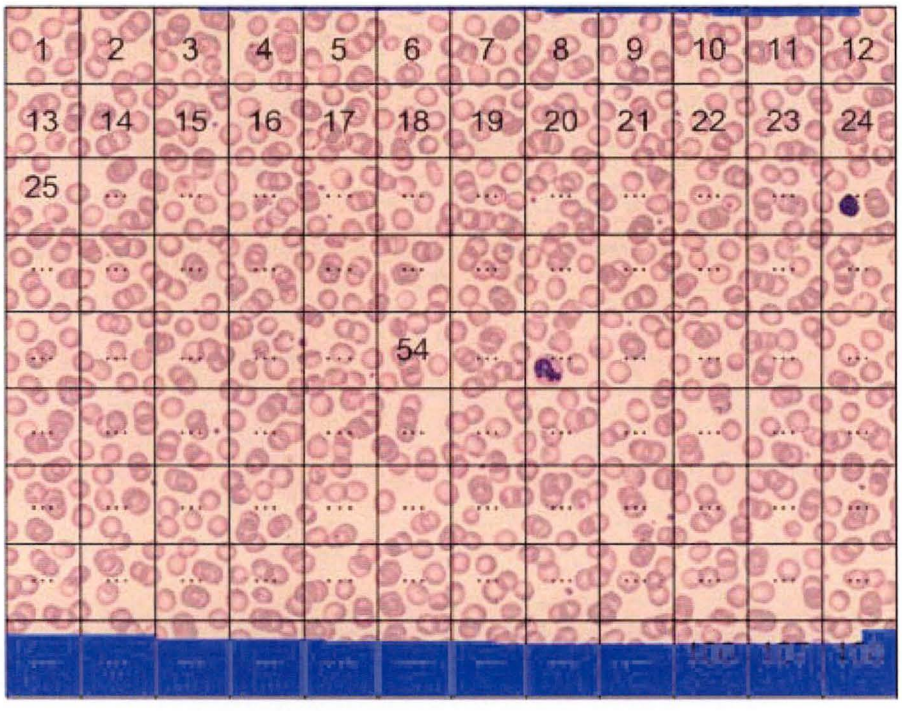


figure 9.1 : méga-image découpée en *tiles*

Une fois que tous les *tiles* composant la méga-image sont découpés et numérotés, ils sont compressés en fonction de la place qu'ils prendront dans la giga-image. En effet, comme nous l'avons vu dans le chapitre 5 de la partie 2, le logiciel jj2000 permet de compresser une image ou un *tile* en fonction de la place que celui-ci prendra dans l'image dans laquelle il sera inséré.

Les 108 tiles de la première méga-image sont donc compressés en fonction de la place qu'ils occuperont dans la giga-image. Le *tile* numéro 1 sera compressé avec le paramètre « ref » fixé à (0,0), le *tile* numéro 2 verra lui son paramètre « ref » fixé à (100,0), pour le *tile* 54, le paramètre sera fixé à (500,500) et le paramètre « ref » du dernier *tile* sera fixé à (1200,0) ainsi de suite pour tous les *tiles*.

→ « ref » = (0,0)

figure 9.2

→ « ref » = (100,0)

figure 9.3

→ « ref » = (500,500)

figure 9.4


→ « ref » = (1200,0)

figure 9.5

Une fois compressé, les différents *tiles* sont sauvegardés sur le disque dur.

Pour la seconde méga-image (figure 9.6), le cheminement est identique à celui utilisé pour la première méga-image, mis à part le fait que lors de la compression des *tiles*, la valeur du paramètre « ref » tient compte du nombre de *tiles* composant la méga-image précédente. En effet, Sur la figure 9.7 représentant la giga-image composée des deux méga-images précédentes, nous voyons que la seconde méga-image se place à droite de la première méga-image. Il faut donc tenir compte du nombre de *tiles* composant la première image lors de la compression des *tiles* de la seconde image.

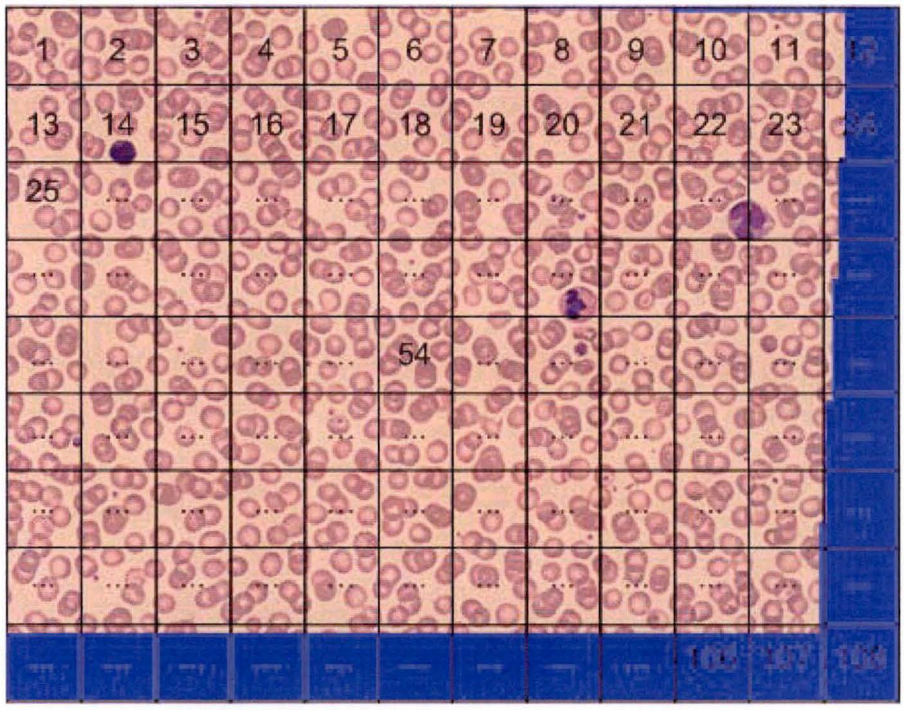


figure 9.6 : Méga-image découpée en *tiles*

Pour donner un exemple, le *tile* numéro 1 de la seconde méga-image sera compressé avec le paramètre fixé à (1200,0), le *tile* numéro 2 sera compressé avec (1300,0), le *tile* 54 avec (1700,500) et ainsi de suite.

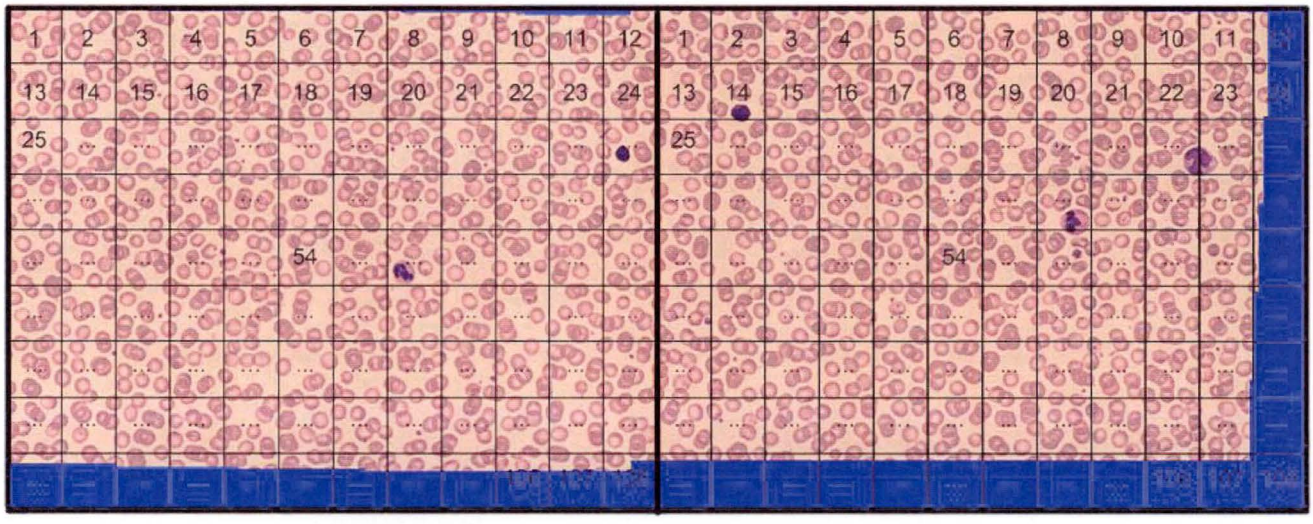


figure 9.7 : Giga-image

9.2 La phase d'assemblage des *tiles*

La phase d'assemblage des *tiles* de la giga-image est la dernière phase intervenant dans la création d'une giga-image. Le but de cette phase est d'assembler tous les *tiles* qui ont été compressés et sauvegardés sur le disque dur lors de la phase de « numérotation, compression et sauvegarde des *tiles* ».

Nous avons vu dans le chapitre 5 de la partie deux, la composition et l'agencement d'un fichier compressé par jpeg2000. Pour rappel, la figure 9.8 illustre la composition d'un tel fichier.

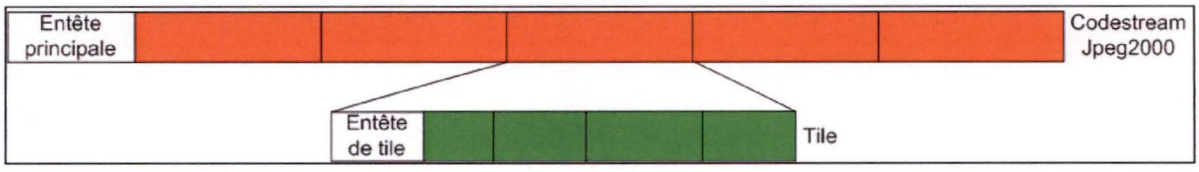


figure 9.8

La figure suivante montre les marqueurs qui composent l'image principale et les *tiles* qui composent celle-ci.

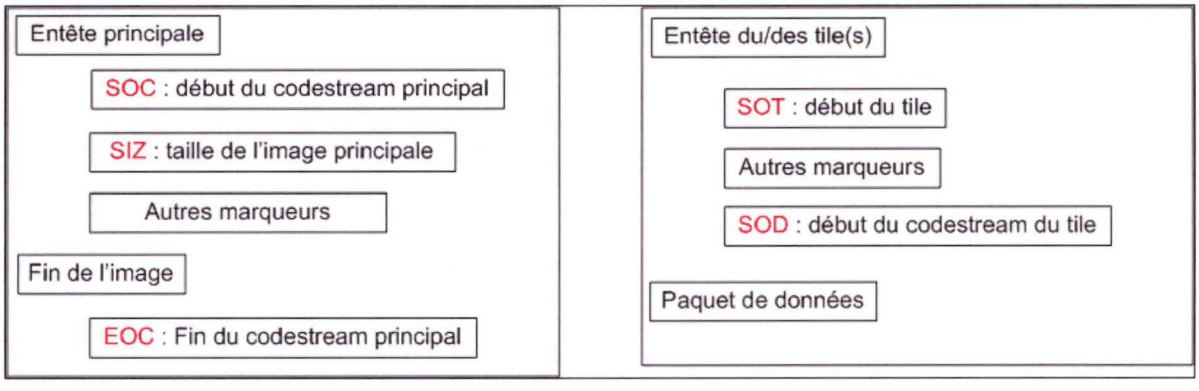


figure 9.9 : Principaux marqueurs

Avant de pouvoir assembler tous les *tiles* il faut créer le canevas qui va les accueillir. Pour cela il faut respecter le format des fichiers Jpeg2000 défini dans la norme. Les

différentes tâches intervenant dans la phase d' « Assemblage des *tiles* » sont reprises sur la figure 9.10. La première étape consiste à créer l'entête principale. Il faut pour cela placer les marqueurs qui composent l'entête au bon endroit. Il faut aussi connaître la dimension de l'image principale ainsi que celle des *tiles* qui vont composer cette image. La deuxième étape consiste à traiter les *tiles*. Les différentes sous-étapes de ce traitement sont reprises par la figure 9.11. Une fois tous les *tiles* traités il ne reste plus qu'à ajouter le marqueur de fin au *codestream* principal.

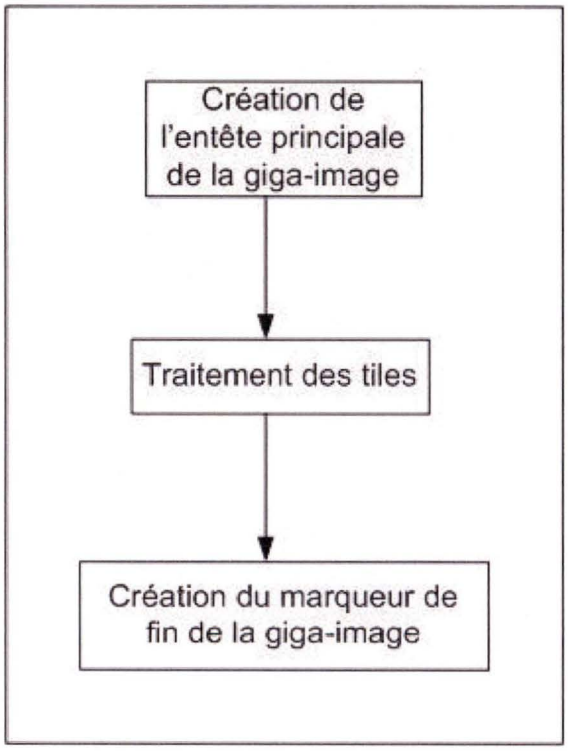


figure 9.10 : Etapes de la phase d'assemblage.

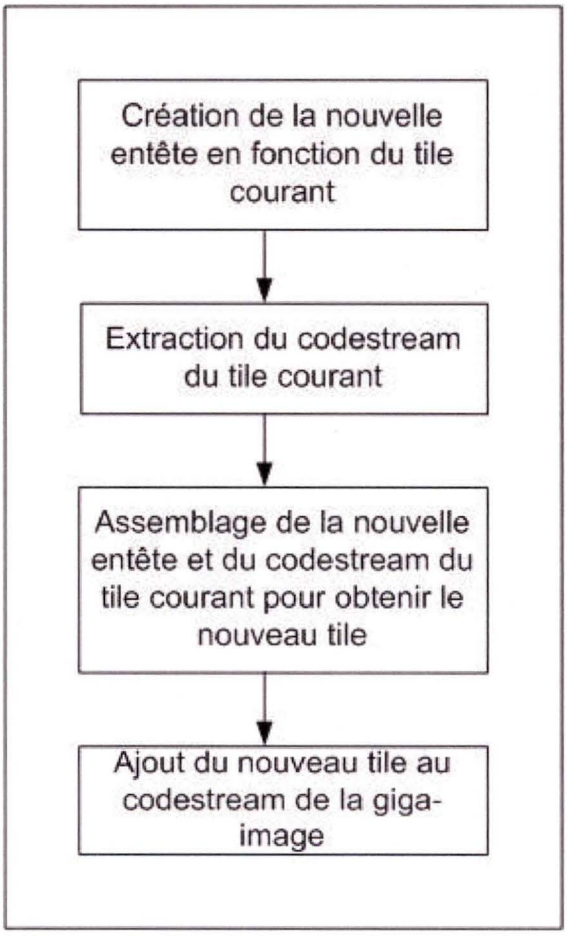


figure 9.11 : Sous-étapes de l'étape de traitement des *tiles*.

La figure suivante montre le lien entre les phases de « création et stockage des méga-images », « numérotation, compression et sauvegarde des *tiles* », et d' « assemblage des *tiles* ».

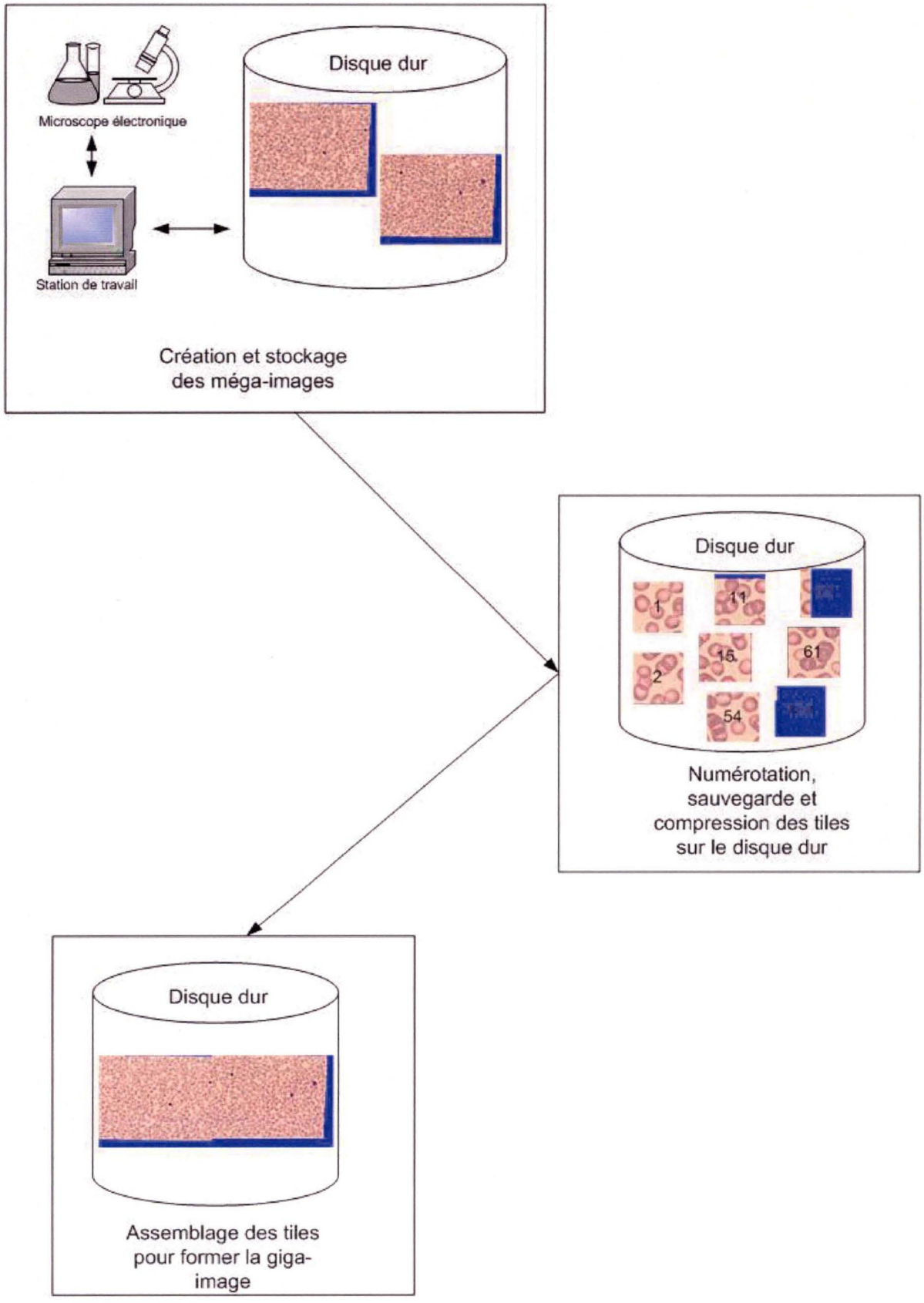


figure 9.12 : Schéma représentant le passage de méga à giga-image

Partie 4 : Discussion

Chapitre 10 : Analyse de la solution

Le but de cette partie discussion est double. D'une part nous allons essayer d'apporter une vision plus critique de l'application qui a été réalisée dans la troisième partie, et d'autre part, nous essayerons de donner des pistes pour solutionner les différents problèmes mis en évidence. Certaines critiques énoncées par la suite sont issues de [Zuy03].

10.1 Le volume élevé de données à manipuler

Comme nous l'avons déjà plusieurs fois mis en évidence tout au long de cet ouvrage, le volume de données à manipuler lors de la réalisation d'un frottis numérique est considérable. Mais nous n'avons jamais relever le problème du temps mis pour réaliser ces frottis numériques. En effet, les différentes phases nécessaires à la réalisation des frottis sont relativement longues. Certaines phases comme l'acquisition des images avec le microscope électronique ne peuvent être réduites ou optimisées au niveau de la durée. Mais les phases comme celles de la coregistration pourraient être optimisées. Une optimisation pourrait notamment se faire au niveau de l'étape concernant la comparaison de l'ensemble des vecteurs avec le vecteur étalon. Cette comparaison pourrait être optimisée en réduisant le nombre de vecteurs à comparer. Cette réduction pouvant se faire suite à une étude des zones de recouvrement. Etant donné le temps nécessaire pour réaliser un frottis numérique, il n'y a pas eu beaucoup de tests effectués, et les tests qui ont été effectués l'ont été avec des méga-images de faible taille.

10.2 La qualité des images réalisées

La netteté des images acquises par le microscope a posé certains problèmes, comme il a été dit dans [Zuy03], ce problème de netteté provient de la platine motorisée. Celle-ci est

légèrement inclinée, ce qui implique que pour un même frottis, le plan focal[24] change suivant les zones du frottis. Ce problème pourrait être solutionné par l'utilisation de l'autofocus, mais celui-ci n'étant pas fiable nous avons préféré nous en passer. Une autre solution aurait consisté à changer la platine existante pour une platine dont l'axe ne serait pas incliné.

10.3 Les défauts de l'interface

L'interface permettant la réalisation des méga-images et des giga-images est une fenêtre intégrée au logiciel Analysis. Elle contient néanmoins quelques lacunes. D'une part elle ne donne aucune limitation à l'utilisateur au niveau du nombre de *snapshots* constituant une méga-images, ni du nombre de méga-images constituant une giga-image, ce qui ne limite pas la taille maximale du frottis réalisable. Ceci pouvant impliquer aussi bien un problème de dépassement de mémoire Ram au niveau de la coregistration des *snapshots* en mémoire, qu'un dépassement de la capacité de stockage du disque dur lors de la sauvegarde des tiles sur le disque dur. Un autre point négatif concerne l'unité de mesure choisie pour réaliser un frottis. L'utilisateur doit indiquer le nombre de *snapshots* en ordonnées et en abscisses qui vont former une méga-images, et ensuite le nombre de méga-images en ordonnées et en abscisses qui vont former les giga-images. Ce choix laissé à l'utilisateur n'est pas très adéquat.

D'autres apports comme une mesure en *mm* des dimensions sur la lame serait sans doute un plus pour l'application. Il conviendrait aussi de donner une estimation du temps nécessaire à la réalisation du frottis ainsi qu'une estimation du volume de l'image compressée. Un dernier problème concerne l'impossibilité de stopper proprement le programme une fois qu'il est lancé. Ce problème provient du fait que le programme Analysis fonctionne en monothread, ce qui rend impossible la réalisation d'une application qui à la fois exécute une fonctionnalité déclenchée à partir d'une interface et qui continue à écouter les

[24] La position de l'objectif pour laquelle une image est nette

actions de l'utilisateur sur cette interface. La seule solution pour arrêter le programme et de tuer le processus lié à Analysis.

Une autre limite de l'application concerne le choix de la zone à numériser, en effet le choix de cette zone se fait manuellement avant de lancer l'application. Seulement rien n'indique à l'utilisateur où se situera l'endroit qu'il vient de choisir sur le frottis par rapport à la zone qui va être numérisée. Il s'agit en fait du centre de la méga-image (0,0). Il serait préférable de faire en sorte que l'utilisateur choisisse une zone qui correspondra par la suite au centre de la giga-image.

Conclusion

Comme nous l'avons déjà mis en évidence, les évolutions technologiques ont permis l'élaboration de nouvelles applications permettant de fournir des services de santé à distance. Ces évolutions ont aussi permis et facilité la consultation et l'échange d'informations entre experts distants. La télémicroscopie, au même titre que la télémédecine, a été développée à travers deux approches : une première approche dite « real time », et une seconde appelée «store and forward». C'est cette seconde approche qui fut choisie pour réaliser la solution présentée dans ce mémoire.

L'application que nous avons développée et qui permet la composition automatique de frottis numériques complets est asynchrone. Ceci est un avantage car cela facilite la consultation et l'échange d'informations entre experts. Cependant, pour donner des résultats suffisamment fiables, une solution asynchrone doit fournir assez d'informations afin de permettre à l'expert d'effectuer un diagnostique correct. Dans notre cas il faut numériser une zone suffisamment grande du frottis ($1cm^2$ à un grossissement 100x) ce qui génère de gros volumes de données. (pour $1cm^2$ cela fait 44.3 Go)

Nous avons donc tenté dans ce mémoire de trouver une solution au problème posé par la coregistration d'images ayant des dimensions hors normes. Cette solution a été présentée à travers une application permettant la composition automatique de frottis numériques complets. Nous avons plus précisément insisté sur la phase consistant à coregistrer de grosses images appelées « méga-images ». Les images résultant de cette phase de coregistration ont été appelées « giga-images »

Dans la deuxième partie de ce mémoire nous avons donné un aperçu de l'importance et de l'intérêt du nouveau standard de compression Jpeg2000, qui grâce à ses nombreuses fonctionnalités dédiées au traitement des grandes images et son taux de compression élevé a été déterminant lors de la réalisation de l'application. Dans cette deuxième partie nous avons également dressé une esquisse des travaux précédemment réalisés dans le domaine de la télémicroscopie à la Clinique Universitaire de Mont Godinne.

Pour conclure, je tiens à dire que ce stage à la Clinique universitaire de Mont Godinne fut très enrichissant. Il m'a permis de me plonger dans le monde passionnant de la cytologie et de constater à quel point l'informatique peut aider, par ses nouvelles technologies et ses

développements à sauver des vies humaines. Ce stage m'a donné également un premier aperçu du travail en projet tel qu'il est souvent réalisé dans la vie professionnelle.

Bibliographie

Ouvrages, Articles et Publications

[Chr00]
C. Christopoulos, A. Skodras et T. Ebrahimi
"JPEG2000 The next generation still image coding system coding system", novembre 2000

[Del01]
V. Della Mea, V. Roberto et C.A. Beltrami
"Visualization issues in Telepathology: the role of the Internet Imaging Protocol", juillet 2001

[Enc97]
Microsoft
"Encyclopédie ENCARTA", 1997

[Gon77]
R.C.Gonzales, P.Wintz
"Digital Image Processing", 1977

[Gor00]
M.J. Gormish et M.W. Marcellin
"The JPEG-2000 Standard", mars 2000

[Gra99]
A.K. Graham, W.M Leong, T. Gahm, O'D McGee1
"Telepathology: Clinical utility and methodology", 1999

[Had97]
M.Hadallah
"Codage des images fixes par une méthode hybride basée sur la QV et les approximations fractales. ", 1997

[Har95]
A.d'Hardancourt
"Fou du multimédia. ", Sybex 1995

[Haz03]
Vincent Hazebroucq
"Rapport sur l'état des lieux, en 2003, de la télémédecine française ", 2003

[Jpeg00]
"ISO/IEC FCD15444-1 - JPEG 2000 Part I Final Committee Draft Version 1.0", mars 2000

[Lam00]
K. Mam. "Telepathology: Making Diagnoses is Overcoming Confinements in Place and Time ", South West Cancert News, Issue4, 2000.

[Leo01]
"Practical applications of internet resources for cost-effective telepathology practice"
W-M Leong

[Mar00]
M.W. Marcellin, M.J. Gormish, A. Bilgin et M.P. Boliek
"An Overview of JPEG-2000", 2000

[Mcg01]
O'D McGee et W.M Leong
"Automated complete slide digitization: a medium for simultaneous viewing by multiple pathologists", 2001

[Mic02]
H. Meurisse, J. Fichefet et B. Chatelain
"Projet de Composition automatique de frottis virtuels ", 2002

[San01]
D. Santacruz, R. Grosbois et T. Ebrahimi
"JPEG 2000, la nouvelle norme pour le codage d'images ", Flash informatique, mars 2001

[Tab96]
K.Tabari, S.Tagma
"Compression d'images animées à très faible débit par la géométrie des fractales. ", 1996

[Tau03]
D. Taubman
"JPEG2000 Compression standard for interactive imaging", mars 2003

[Tza]
A. Tzannes et T. Ebrahimi
"JPEG 2000 for Medical Imaging Applications"

Mémoires :

[Geo02] B. Georges. "La télémicroscopie en cytologie hématologique", Mémoire en informatique, FUNDP Namur, 2002

[Zuy03] L. Zuyderhoff. "Composition automatique de frottis numériques", Mémoire en informatique, FUNDP, Namur, 2003

[Kad99] C. Kaddour, "Compression des images fixes par fractals basée sur la triangulation de Delannay et la quantification vectorielle", Mémoire en informatique, université des sciences et de la technologie Houari Boumediene, Algérie, 1999

[Mer00] B. Mercier, "Mise au point d'une station de télémedecine pour le laboratoire d'hématologie ", Mémoire en biologie médicale, Institut Paul Lambin, UCL, 2000.

Sites Internet :

[WebCcm]
"Encyclopédie informatique libre"
http://www.commentcamarche.net

[WebJJ2000]
"JJ2000: An implementation of the JPEG2000 standard in Java"
http://jj2000.epfl.ch/

[WebJpeg2000-1]
"LA COMPRESSION D'IMAGES PAR ONDELETTES (JPEG 2000) "
http://www.image-etc.com/faq/wavelet/Page1.htm

[WebJpeg2000-2]
"Première approche de JPEG 2000"
http://www.ruses.com/Pages/0001000K.htm

[WebJpeg2000-3]
"JPEG 2000: The Next Compression Standard using wavelet technology"
http://www.gvsu.edu/math/wavelets/student_work/EF/index.html

[WebJpeg2000-4]
"JPEG2000 – Our new standard"
http://www.jpeg.org/jpeg2000/index.html

[WebJpeg2000-5]
"Quantification"
http://www.tcom.ch/Tcom/Laboratoires/digivox2000/chap/chap2/quantification.htm

[WebOnd]
" TIPE : Compression par ondelettes"
http://donut.99.free.fr/En-vrac/tipe/ondelettes.htm

[WebTelem-1]
"Etat de l'art et apport de la Télémédecine en Hospitalisation A Domicile"
F. de Montbel, C. Martageix et B. Saille
http://www.utc.fr/~farges/dess_tbh/01_02/Projets/telhad/telhad.html

[WebTelem-2]
"Télémédecine et Imagerie Médicale"
http://www.ese-metz.fr/metz/eleves/themes/imagerie/tele_part1.html

[WebTelem-3]
"Advanced Networking for Telemicroscopy"
http://www.isoc.org/inet2000/cdproceedings/5a/5a_4.htm

[WebTelem-4]
"Telemicroscopy over Internet"

http://ipath.sourceforge.net/?q=node/view/5
[WebTelem-5]
"Zem Technology - Platform independent static and dynamic telemicroscopy"
http://www.zem.com/telemicroscopy.html

[WebTelep]
"Store-and-forward Telepathology"
http://www.telemed.uniud.it/FVG/telstat.html

Annexes

Annexe 1 : Le codage d'image RGB et YUV. (transformée couleur)

Issu de « http://www.commentcamarche.net/video/couleur.php3 »

Le codage RGB

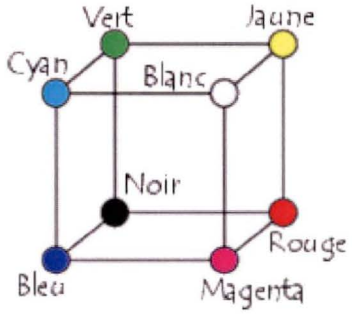
Le codage RGB, mis au point en 1931 par la Commission Internationale de l'Eclairage (CIE) consiste à représenter l'espace des couleurs à partir de trois rayonnements monochromatiques de couleurs :

- rouge (de longueur d'onde égale à 700,0 nm),
- vert (de longueur d'onde égale à 546,1 nm),
- bleu (de longueur d'onde égale à 435,8 nm).

Cet espace de couleur correspond à la façon dont les couleurs sont généralement codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs.

Ainsi, le modèle RGB propose de coder sur un octet chaque composante de couleur, ce qui correspond à 256 intensités de rouge (28), 256 intensités de vert et 256 intensités de bleu, soient 16777216 possibilités théoriques de couleurs différentes, c'est-à-dire plus que ne peut en discerner l'oeil humain (environ 2 millions). Toutefois, cette valeur n'est que théorique car elle dépend fortement du matériel d'affichage utilisé.

Etant donné que le codage RGB repose sur trois composantes proposant la même gamme de valeur, on le représente généralement graphiquement par un cube dont chacun des axes correspond à une couleur primaire :



Le codage YUV

Le modèle YUV (appelé aussi CCIR 601) est un modèle de représentation de la couleur dédié à la vidéo analogique. Il s'agit du format utilisé dans les standards PAL (Phase Alternation Line) et SECAM (Séquentiel Couleur avec Mémoire). Le paramètre Y représente la luminance (c'est-à-dire l'information en noir et blanc), tandis que U et V permettent de représenter la chrominance, c'est-à-dire l'information sur la couleur. Ce modèle a été mis au point afin de permettre de transmettre des informations colorées aux téléviseurs couleurs, tout en s'assurant que les téléviseurs noir et blanc existant continuent d'afficher une image en tons de gris.

Voici les relations liant Y à R, G et B, U à R et à la luminance, et enfin V à B et à la luminance :

- $Y = 0.299R + 0.587\ G + 0.114\ B$
- $U = -0.147R - 0.289\ G + 0.463B = 0.492(B - Y)$
- $V = 0.615R - 0.515G - 0.100B = 0.877(B-Y)$

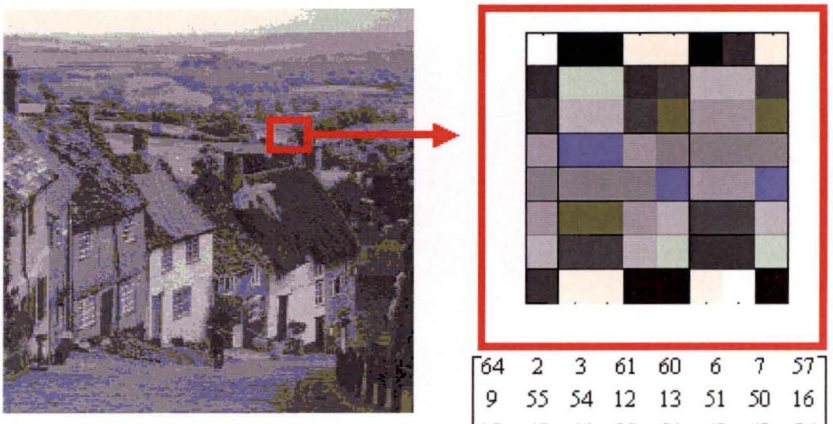
Ainsi U est parfois noté Cr et V noté Cb, d'où la notation YCrCb.

Annexe 2 : La transformée en ondelettes. (Haar)

Issu de "http://aix1.uottawa.ca/~jkhoury/haarf.htm" rédigé par Joseph Khoury

Introduction Une fois recherchées de l'Internet, les images digitales prennent un temps considérable pour télécharger et utilisent une grande proportion de mémoire de l'ordinateur. Les **ondelettes de Haar** que nous discuterons dans cette application représentent une façon de comprimer des images digitales de manière qu'elles prennent moins d'espace une fois stockées.

L'idée fondamentale derrière cette méthode de compression est de traiter une image digitale comme un arrangement rectangulaire des nombres ou une matrice. Chaque image se compose d'un nombre assez grand de petits carrés appelés **Pixels** (Picture Elements en anglais). La matrice correspondant à une image digitale associe un nombre entier à chaque Pixel. Par exemple, une image grise de dimension 256x256, est stockée comme une matrice du format 256x256, avec chaque élément de la matrice est un nombre entier entre 0 (pour le noir) à 225 (pour le blanc). La technique de compression de JPEG divise une image en blocs 8x8 et assigne une matrice à chaque bloc. On peut employer quelques techniques d'algèbre linéaire pour maximiser la compression de l'image et en même temps maintenir un niveau acceptable de détail.



Les images sont composées des Pixels représentés par des nombres

$$\begin{bmatrix} 64 & 2 & 3 & 61 & 60 & 6 & 7 & 57 \\ 9 & 55 & 54 & 12 & 13 & 51 & 50 & 16 \\ 17 & 47 & 46 & 20 & 21 & 43 & 42 & 24 \\ 40 & 26 & 27 & 37 & 36 & 30 & 31 & 33 \\ 32 & 34 & 35 & 29 & 28 & 38 & 39 & 25 \\ 41 & 23 & 22 & 44 & 45 & 19 & 18 & 48 \\ 49 & 15 & 14 & 52 & 53 & 11 & 10 & 56 \\ 8 & 58 & 59 & 5 & 4 & 62 & 63 & 1 \end{bmatrix}$$

Transformations des vecteurs par les Ondelettes de Haar Avant d'expliquer la transformation d'une matrice, voyons tout d'abord comment les ondelettes transforment les vecteurs (lignes d'une matrice). Supposez que

$$r = [420 \quad 680 \quad 448 \quad 708 \quad 1260 \quad 1420 \quad 1600 \quad 1600]$$

est une ligne d'une matrice 8x8. En géneral, si le vecteur a 2^k composantes, alors le processus de transformation consiste en k étapes. Dans notre cas, le vecteur a $8=2^3$ composantes, et par suite il y aura trois étapes.
Nous effectuons les opérations suivantes sur les entrées du vecteur r:

1. Divisez les entrées de r en quatre paires :
   (420, 680), (448, 708), (1260, 1410), (1600, 600).

2. Formez la moyenne de chacune de ces paires:

$$\frac{420+680}{2}=550,\quad \frac{448+708}{2}=578,\quad \frac{1260+1420}{2}=1340,\quad \frac{1600+1600}{2}=1600$$

Ceux-ci formeront les quatre premières entrées du prochain vecteur r_1.

3. Soustrayez chaque moyenne de la première entrée de la paire pour obtenir les nombres:

$$-130,\quad -130,\quad -75,\quad 0.$$

Ceux-ci formeront les quatre premières entrées du prochain vecteur r_1.

4. Formez le nouveau vecteur:

$$r_1=[550\quad 578\quad 1340\quad 1600\quad -130\quad -130\quad -80\quad 0].$$

Notez que le vecteur r_1 peut être obtenu à partir de r en multipliant r à droite par la matrice:

$$W_1 = \begin{bmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & -1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 & -1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 0 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & -1/2 \end{bmatrix}$$

Les quatre premières entrées de r_1 s'appellent les **coefficients d'approximation** et les quatre dernières entrées s'appellent les **coefficients de détail.**

Pour notre prochaine étape, nous regardons les quatre premières entrées de r_1 en tant que deux paires que nous prenons leurs moyennes comme dans l'étape 1 ci-dessus. Ceci donne les deux premières entrées: 564, 1470 du nouveau vecteur r_2. Ce sont nos nouveaux coefficients d'approximation. La troisième et la quatrièmes entrée de r_2 sont obtenues en soustrayant ces moyennes du premier élément de chaque paire. Ceci donne les nouveaux coefficients de détail: -14, -130. Les quatre dernières entrées de r_2 sont identiques aux coefficients de détail de r_1:

$$r_2 = [564 \quad 1470 \quad -14 \quad -130 \quad -130 \quad -130 \quad -80 \quad 0]$$

Ici, le vecteur r_2 peut être obtenu de r_1 en multipliant à droite par la matrice :

$$W_2 = \begin{bmatrix} 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & -1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Pour la dernière étape, faites la moyenne des deux premières entrées de r_2, et comme avant, soustrayez la réponse de la première entrée. Ceci a comme conséquence le vecteur suivant:

$$r_3 = [1017 \quad -453 \quad -14 \quad -130 \quad -130 \quad -130 \quad -80 \quad 0]$$

Comme avant, le vecteur r_3 peut être obtenu de r_1 en multipliant à droite par la matrice :

$$W_3 = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Par conséquent, on obtient r_3 immédiatement de r en utilisant l'équation suivante :

$$r_3 = W_1 W_2 W_3 r$$

Posons

$$W = W_1 W_2 W_3 = \begin{bmatrix} 1/8 & 1/8 & 1/4 & 0 & 1/2 & 0 & 0 & 0 \\ 1/8 & 1/8 & 1/4 & 0 & -1/2 & 0 & 0 & 0 \\ 1/8 & 1/8 & -1/4 & 0 & 0 & 1/2 & 0 & 0 \\ 1/8 & 1/8 & -1/4 & 0 & 0 & -1/2 & 0 & 0 \\ 1/8 & -1/8 & 0 & 1/4 & 0 & 0 & 1/2 & 0 \\ 1/8 & -1/8 & 0 & 1/4 & 0 & 0 & -1/2 & 0 \\ 1/8 & -1/8 & 0 & -1/4 & 0 & 0 & 0 & 1/2 \\ 1/8 & -1/8 & 0 & -1/4 & 0 & 0 & 0 & -1/2 \end{bmatrix}.$$

Noter les faits suivant:

- Les colonnes de la matrice W_1 forment un sous-ensemble **orthogonal** de R^8 (l'espace vectoriel de dimension 8 sur R); en d'autres mots, ces colonnes sont deux à deux orthogonaux (calculer leurs produits scalaires). Elles forment en particulier une base R^8. Par conséquent, W_1 est inversible. De même, W_2 et W_3 sont inversibles.
- Comme un produit des matrices inversible, W est elle-même inversible, et ses colonnes forment une base orthonormale de R^8. L'inverse de W est donnée par :

$$W^{-1} = W_3^{-1} W_2^{-1} W_1^{-1}$$

Le fait que W est inversible nous permet de rechercher notre image de la forme comprimée en utilisant la relation :

$$r = W^{-1} r_3.$$

Supposez que A est la matrice correspondante à une certaine image. La transformation de Haar consiste à faire la moyenne et la différence de chaque ligne de la matrice A et puis répéter les mêmes opérations sur les colonnes de la matrice résultante. Les transformations sur les lignes donnent la matrice AW. La matrice obtenue en faisant la moyenne et différence sur les colonnes de AW est obtenue en multipliant AW à gauche par la matrice W^{T} (la transposée de W). Ainsi, la transformation de Haar ``stocke`` une matrice comme $W^{\mathrm{T}}AW$. Soit S la matrice transformée:

$$S = W^T AW.$$

En utilisant les propriétés des inverses matricielles, on peut retrouver la matrice originale :

$$A = (W^T)^{-1} S W^{-1} = (W^{-1})^T S W^{-1}.$$

Ceci nous permet de voir l'image originale (décomprimant l'image comprimée).

Il est temps maintenant pour un exemple.

Exemple Supposez que nous avons une image 8x8 représentée par la matrice :

$$A = \begin{bmatrix} 576 & 704 & 1152 & 1280 & 1344 & 1472 & 1536 & 1536 \\ 704 & 640 & 1156 & 1088 & 1344 & 1408 & 1536 & 1600 \\ 768 & 832 & 1216 & 1472 & 1472 & 1536 & 1600 & 1600 \\ 832 & 832 & 960 & 1344 & 1536 & 1536 & 1600 & 1536 \\ 832 & 832 & 960 & 1216 & 1536 & 1600 & 1536 & 1536 \\ 960 & 896 & 896 & 1088 & 1600 & 1600 & 1600 & 1536 \\ 768 & 768 & 832 & 832 & 1280 & 1472 & 1600 & 1600 \\ 448 & 768 & 704 & 640 & 1280 & 1408 & 1600 & 1600 \end{bmatrix}$$

Les transformations sur les lignes de A donne la matrice :

$$L = AW = \begin{bmatrix} 1200 & -272 & -288 & -64 & -64 & -64 & -64 & 0 \\ 1185 & -288 & -225 & -96 & 32 & 34 & -32 & -32 \\ 1312 & -240 & -272 & -48 & -32 & -128 & -32 & 0 \\ 1272 & -280 & -160 & -16 & 0 & -192 & 0 & 32 \\ 1256 & -296 & -128 & 16 & 0 & -128 & -32 & 0 \\ 1272 & -312 & -32 & 16 & 32 & -96 & 0 & 32 \\ 1144 & -344 & -32 & -112 & 0 & 0 & -96 & 0 \\ 1056 & -416 & -32 & -128 & 160 & 32 & -64 & 0 \end{bmatrix}$$

En prenant les moyennes et les différences sur les colonnes de of L, on obtient la matrice suivante :

$$S = W^T L = \begin{bmatrix} 1212 & -306 & -146 & -54 & -24 & -68 & -40 & 4 \\ 30 & 36 & -90 & -2 & 8 & -20 & 8 & -4 \\ -50 & -10 & -20 & -24 & 0 & 72 & -16 & -16 \\ 82 & 38 & -24 & 68 & 48 & -64 & 32 & 8 \\ 8 & 8 & -32 & 16 & -48 & -48 & -16 & 16 \\ 20 & 20 & -56 & -16 & -16 & 32 & -16 & -16 \\ -8 & 8 & -48 & 0 & -16 & -16 & -16 & -16 \\ 44 & 36 & 0 & -8 & 80 & -16 & -16 & 0 \end{bmatrix}$$

L'idée de la transformation de Haar est que les secteurs de la matrice originale qui contient peu de variation finiront d'être nuls dans la matrice transformée. Une matrice est considérée **clairsemée** si elle a une proportion élevée d'entrées nulles. Les matrices clairsemées prennent beaucoup moins de mémoire une fois stockées. Puisque nous ne pouvons pas nous attendre à

ce que les matrices transformées toujours soient clairsemées, nous décidons d'une valeur-seuil non négative connue comme ε, et nous laissons ensuite n'importe quelle entrée dans la matrice transformée dont la valeur absolue est moins que ε d'être remis à zéro. Ceci nous laissera avec un genre de matrice clairsemée. Si ε est zéro, toutes les entrées de la matrice demeurent inchangées.

Chaque fois que vous cliquez une image pour la télécharger de l'Internet, l'ordinateur de source rappelle la matrice transformée de sa mémoire. Il envoie d'abord les coefficients globaux d'approximation et de plus grands coefficients de détail et un peu plus tard les coefficients plus petits de détail. Pendant que votre ordinateur reçoit l'information, il commence à reconstruire progressivement les détails jusqu'à ce que l'image originale soit entièrement reconstruite.

L'algèbre linéaire peut rendre le processus de compression plus rapide, plus efficace

Commençons par rappeler qu'une matrice carrée de format *nxn* s'appelle orthogonale si ses colonnes forment une base orthonormale de R^n. En d'autres mots, les colonnes de A sont deux à deux orthogonales et la longueur de chaque vecteur colonne est 1. D'une manière équivalente, A est orthogonal si son inverse est égal à sa transposée. Cette dernière propriété rend le processus de rechercher l'image transformée par l'intermédiaire de l'équation

$$A = (W^T)^{-1} S W^{-1} = (W^{-1})^T S W^{-1} = W S W^T$$

beaucoup plus rapide.

Une autre propriété puissante des matrices orthogonales est qu'elles préservent la grandeur. En d'autres termes, si v est un vecteur de R^n et A est une matrice orthogonale, alors $\|Av\|=\|v\|$. Voici comment cela fonctionne :

$$\begin{aligned} \| Av \|^2 &= (Av)^T (Av) \\ &= v^T A^T A v \\ &= v^T I v \\ &= v^T v \\ &= \| v \|^2 \end{aligned}$$

Ceci montre $\|Av\|=\|v\|$. Aussi, la transformation qui utilise une matrice orthogonale préserve l'angle: rappelez-vous que le cosinus de l'angle entre deux vecteurs u et v est donné par:

$$\cos \phi = \frac{u.v}{\| u \| \| v \|}$$

Si A est une matrice orthogonale et ψ est l'angle entre les deux vecteurs Au et Av, alors

$$\begin{aligned}
\cos\psi &= \frac{(Au).(Av)}{\|Au\|\|Av\|} \\
&= \frac{(Au)^T(Av)}{\|u\|\|v\|} \\
&= \frac{u^T A^T Av}{\|u\|\|v\|} \\
&= \frac{u^T v}{\|u\|\|v\|} \\
&= \frac{u.v}{\|u\|\|v\|} \\
&= \cos\phi
\end{aligned}$$

Comme la grandeur et l'angle sont préservés, il y a sensiblement moins de déformation produite dans l'image reconstruite quand une matrice orthogonale est employée. Puisque la matrice W de transformation est le produit de trois autres matrices, on peut normaliser W en normalisant chacune des trois matrices. La version normale de W est :

$$W = \begin{bmatrix}
\sqrt{8}/64 & \sqrt{8}/64 & 1/2 & 0 & \sqrt{2}/4 & 0 & 0 & 0 \\
\sqrt{8}/64 & \sqrt{8}/64 & 1/2 & 0 & -\sqrt{2}/4 & 0 & 0 & 0 \\
\sqrt{8}/64 & \sqrt{8}/64 & -1/2 & 0 & 0 & \sqrt{2}/4 & 0 & 0 \\
\sqrt{8}/64 & \sqrt{8}/64 & -1/2 & 0 & 0 & -\sqrt{2}/4 & 0 & 0 \\
\sqrt{8}/64 & -\sqrt{8}/64 & 0 & 1/2 & 0 & 0 & \sqrt{2}/4 & 0 \\
\sqrt{8}/64 & -\sqrt{8}/64 & 0 & 1/2 & 0 & 0 & -\sqrt{2}/4 & 0 \\
\sqrt{8}/64 & -\sqrt{8}/64 & 0 & -1/2 & 0 & 0 & 0 & \sqrt{2}/4 \\
\sqrt{8}/64 & -\sqrt{8}/64 & 0 & -1/2 & 0 & 0 & 0 & -\sqrt{2}/4
\end{bmatrix}$$

Remarque Si vous regardez de plus proche le processus que nous avons décrit ci-dessus, vous noterez que la matrice W n'est rien qu'une matrice de changement de base pour R^8. En d'autres termes, les colonnes de W forment une nouvelle base (``très utile'') de $R^8$. Ainsi quand vous multipliez un vecteur $v$ (écrit dans la base standard) de $R^8$ par $W$, vous obtenez les coordonnées de $v$ dans cette nouvelle base. Certaines de ces coordonnées peuvent être ``négligées'' en utilisant notre seuil et ceci permet à la matrice transformée d'être plus facile et plus rapide à stocker.

La valeur- seuil Si nous choisissons notre valeur- seuil ε de sorte qu'elle soit strictement positive (non-nulle), alors quelques entrées de la matrice transformée seront remises à zéro et donc un certain détail sera perdu quand l'image est décomprimée. La question clé est alors comment choisir ε de sorte que la compression soit faite efficacement avec une déformation minimale à l'image. Nous définissons le **rapport de compression** comme étant le rapport des

entrées non-nulles dans la matrice transformée ($S=W^TAW$) au nombre d'entrées non-nulles dans la matrice comprimée obtenue à partir de S en appliquant le seuil ε.

Annexe 3 : Le contenu du Codestream Jpeg2000.

Issu de **ISO/IEC FCD15444-1 : 2000 (V1.0, 16 March 2000)**

Codestream syntax

(This annex forms an integral part of this Recommendation | International Standard)

This Annex specifies the marker and marker segment syntax defined by this Recommendation | International Standard.

These markers and marker segments provide codestream information for this Recommendation | International Standard.

Further, this Annex provides a marker and marker segment syntax that is designed to be used in future specifications that include this Recommendation | International Standard as a normative reference.

This Recommendation | International Standard does not include a definition of compliance or conformance. The parameter values of the syntax described in Annex A are not intended to portray the capabilities required to be compliant.

A.1 Headers and marker segments

This Recommendation | International Standard uses marker segments to delimit and signal the characteristics of the codestream. This set of markers and marker segments is the minimal information needed to achieve the features of this Recommendation | International Standard and is not a file format. A complete file format is offered in Annex I.

Headers are collections of markers and marker segments. There are two types of headers in this specification. The main header is found at the beginning of the codestream. The tile-part headers are found at the beginning of each tile-part (see below). Some markers and marker segments are restricted to only one of the two types of headers while others can be found in either.

A.1.1 Markers and marker segments

Every marker is two bytes long. The first byte consists of a single 0xFF byte. The second byte denotes the specific marker and can have any value in the range 0x01 to 0xFE. Many of these markers are already used in ITU-T Rec. T.81 | ISO/IEC 10918-1 and ITU-T Rec. T.84 | ISO/IEC 10918-3 and shall be regarded as reserve unless specifically used.

A marker segment includes a marker and associated parameters, called marker parameters. In every marker segment the first two bytes after the marker shall be an unsigned big endian integer value that denotes the length in bytes of the marker parameters (including two bytes of this length parameter but not the two bytes of the marker itself).

A.1.2 Types of markers and marker segments

Six types of markers and marker segments are used: delimiting, fixed information, functional, in bit stream, pointer, and informational. Delimiting marker and marker segments must be used to frame the headers and the data. Fixed information marker segments give required information about an image. The location of these marker segments, like delimiting marker segments, is specified. Functional marker segments are used to describe the coding functions used. In bit stream markers and marker segments are used for error resilience. Pointer marker segments point to specific offsets in the bit stream. Informational marker segments provide ancillary information.

A.1.3 Syntax similarity with ITU-T Rec. T.81 | ISO/IEC 10918-1

The marker and marker segment syntax uses the same construction as defined in ITU-T Rec.T.81 | ISO/IEC 10918-1.
Some of the markers are exactly the same. Those that are not have numbers that were reserved in ITU.T Rec. 81 | IS 10918-1, ITU-T Rec. T.84 | ISO/IEC 10918-3, and ITU-T Rec. T.87 | ISO/IEC 14495-1 are registered by the registration process defined in ITU-T Rec. T.86 | ISO/IEC 10918-4.

The marker range 0xFF30 — 0xFF3F is reserved by this specification for markers without marker parameters. This will enable backward compatibility. Table A-1 shows in which specification these markers and marker segments are defined.

Table A-1 — Marker definitions

Marker value range	Standard definition
0xFF00, 0xFF01, 0xFFFE, 0xFFC0 — 0xFFDF	Defined in ITU-T Rec. T.81 I ISO/IEC 10918-1
0xFFF0 — 0xFFF6	Defined in ITU-T Rec. T.84 I ISO/IEC 10918-3
0xFFF7 — 0xFFF8	Defined in ITU-T Rec. T.87 I ISO/IEC 14495-1
0xFF4F — 0xFF6F, 0xFF90 — 0xFF93	Defined in this International Standard I Recommendation
0xFF30 — 0xFF3F	Reserved for definition as markers only (no marker segments)

A.1.4 Marker and marker segment and codestream rules

— Marker segments, and therefore the headers, are a multiple of 8 bits (one byte). Further, the bit stream data between the headers are padded to also be aligned to a multiple of 8 bits.
— All markers and marker segments in a tile-part header apply only to the tile to which it belongs.
— All markers and marker segments in the main header apply to the whole image unless specifically overridden by marker segments in a tile-part header.
— Delimiting and fixed information marker segments must appear at specific points in the codestream.
— The marker segments shall correctly describe the image as represented by the codestream. If truncation, alteration, or editing of the codestream has been performed, the marker segments shall be updated accordingly.
— All parameter values in marker segments are big endian (most significant byte first).
— All markers with the marker value between 0xFF30 and 0xFF3F have no marker parameters.
NOTE — The markers the range 0xFF30 — 0xFF3F may be used by future extensions. They may or may not be skipped by a decoder without ramification.

A.1.5 Key to graphical descriptions (informative)

Each marker segment is described in terms of its function, usage, and length. The function describes the information contained in the marker segment. The usage describes the logical location and frequency of this marker segment in the codestream. The length describes which parameters determine the length of the marker segment.

These descriptions are followed by a figure that shows the order and relationship of the parameters in the marker segment. Figure A-1 shows an example of this type of figure. The marker segments are designated by a three letter abbreviation. The parameter values have capital letter designations with the marker's abbreviation as a subscript. A rectangle is used to indicate the parameters in the marker segment. The width of the rectangle is proportional to the number of bytes in the field. A shaded rectangle (diagonal stripes) indicates that the parameter is of varying size. Two parameters with superscripts and a gray area between indicate a run of several of these parameters.

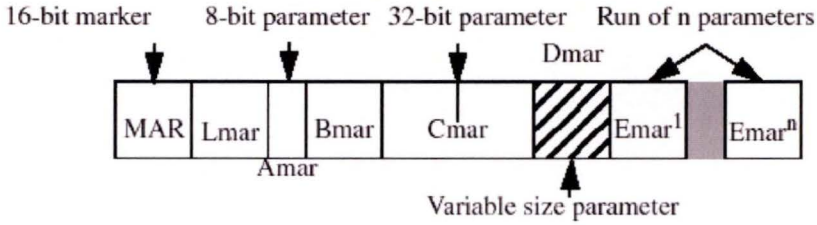


Figure A-1 — Example of the marker segment description figures

The figure is followed by a list that describes the meaning of each parameter in the marker segment. If parameters arerepeated, the length and nature of the run of parameters is defined. As an example, in Figure A-1, the first rectangle represents the marker with the name MAR. The second rectangle represents the length parameter. Parameters Amar, Bmar, Cmar, and Dmar are 8, 16, 32 bit and variable length respectively. The parameter Emari has a run from 1 to n.

After the list is a table that either describes the allowed parameter values or provides references to other tables that describe these values. Tables for individual parameters are provided to describe any parameter without a simple numerical value. In some cases theses parameters are described by a bit value in a bit field. In this case, the bits that do not matter for this parameter are denoted with an "x."

Some marker segments are described using the notation "Sxxx" and "SPxxx" (for a marker named XXX). The Sxxx parameter selects between many possible states of the SPxxx parameter. According to this selection, the SPxxx parameter or parameter list is modified.

A.2 Information in the marker segments

Table A-2 lists the markers specified in this Recommendation | International Standard. Table A-3 shows a list of the information provided by the syntax and which marker segment contains that information.

Table A-2 — List of marker segments

	Name	Code	Main header[a]	Tile-part header[a]
Delimiting marker segments				
Start of codestream	SOC	0xFF4F	required	not allowed
Start of tile-part	SOT	0xFF90	not Allowed	required
Start of data	SOD	0xFF93	not allowed	last marker
End of codestream[b]	EOC	0xFFD9	not allowed	not allowed
Fixed information marker segments				
Image and tile size	SIZ	0xFF51	required	not allowed
Functional marker segments				
Coding style default	COD	0xFF52	required	optional
Coding style component	COC	0xFF53	optional	optional

Table A-2 — List of marker segments

	Name	Code	Main header[a]	Tile-part header[a]
Region-of-interest	RGN	0xFF5E	optional	optional
Quantization default	QCD	0xFF5C	required	optional
Quantization component	QCC	0xFF5D	optional	optional
Progression order default	POD	0xFF5F	optional[c]	optional[c]
Pointer marker segments				
Tile-part lengths, main header	TLM	0xFF55	optional	not allowed
Packet length, main header	PLM	0xFF57	optional	not allowed
Packet length, tile-part header	PLT	0xFF58	not allowed	optional
Packed packet headers, main header	PPM	0xFF60	optional[d]	not allowed
Packed packet headers, tile-part header	PPT	0xFF61	not allowed	optional[d]
In bit stream marker segments				
Start of packet	SOP	0xFF91	not allowed	optional, in bit stream
End of packet header	EPH	0xFF92	not allowed	optional, in bit stream
Informational marker segments				
Comment and extension	CME	0xFF64	optional	optional

a. Required means the marker segment shall be in this header, optional means it may be used.
b. The EOC marker is the last in the codestream. It is in neither the main nor the tile-part headers.
c. The POD marker segment is required if there are progression order changes.
d. Either the PPM or PPT marker segment is required if the packet headers are not distribuited in the bit stream. If the PPM marker segment is used then PPT marker segments shall not be used, and visa versa.

Table A-3 — Information in the marker segments

Information	Marker segment
Capabilities Image size or reference grid size (height and width) Tile size (height and width) Number of components Component transform used Component precision Component mapping to the reference grid (sub-sampling)	SIZ
Tile number Tile-part data length	SOT, TLM

Table A-3 — Information in the marker segments

Information	Marker segment
Coding style Number of decomposition levels Progression order Number of layers Code-block size Code-block style Wavelet transform	COD, COC
Region of interest shift	RGN
No quantization Quantization implicit Quantization explicit	QCD, QCC
Progression starting point Progression ending point Progression order default	POD
Error resilience End of packet header	SOP, EPH
Code-block values for new layers Code-block layer number Code-block inclusion Maximum bit depth Truncation point Bit stream length for decomposition level and layer in a code-block	packet header, PPM, PPT
Packet lengths	PLM, PLT
Optional information	CME

A.3 Construction of the codestream

Figure A-2 shows the construction of the codestream. Figure A-3 shows the main header construction. Note that all of the solid lines show required marker segments. The marker segments on the left are required to be in a specific location. The dashed lines show optional or possibly not required marker segments. Figure A-4 shows the construction of a tile-part header.

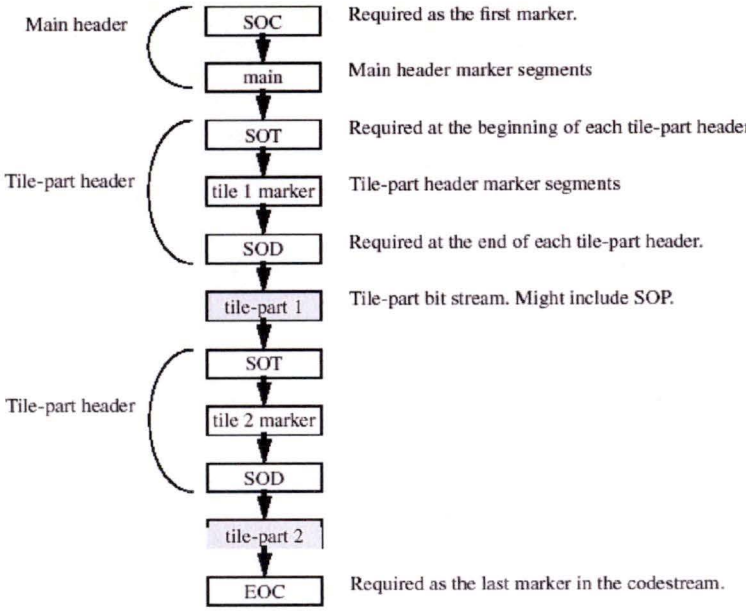


Figure A-2 — Construction of the codestream

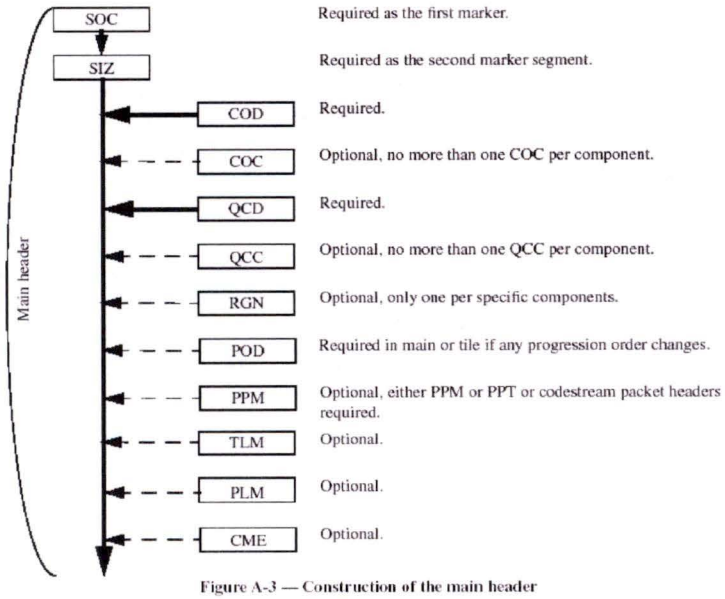


Figure A-3 — Construction of the main header

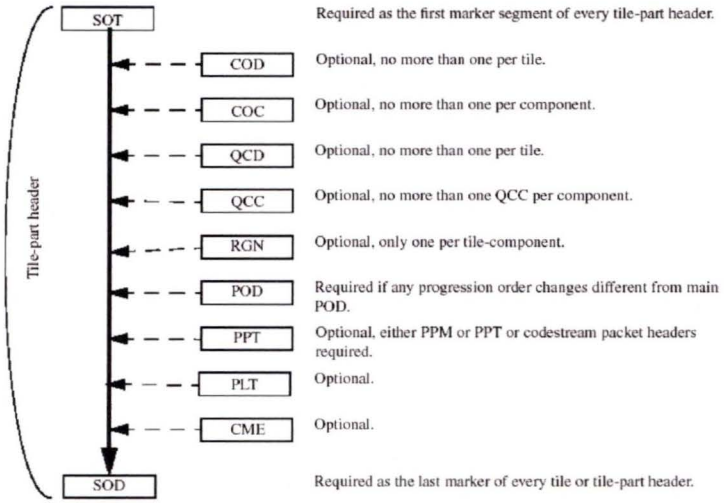


Figure A-4 — Construction of a tile-part header

The COD and COC marker segments and the QCD and QCC marker segments have hierarchy of usage. This is designed to allow tile-components to have dissimilar coding and quantization characteristics with a minimum of signalling.

For example, the COD marker segment is required in the main header. If all components in all the tiles are coded the same way, this is all that is required. If there is one component that is coded differently than the others (for example the luminance component of an image composed of luminance and chrominance components) then the COC can denote that in the main header. If one or more components are coded differently in different tiles, then the COD and COC are used in a similar manner to denote this in the tile-part headers.

The POD marker likewise may appear in the main header, and is used in all tiles, unless a different POD appears in the tile header.

A.4 Delimiting markers

The delimiting marker segments shall be present in all codestreams conforming to this Recommendation | International Standard. Each codestream has only one SOC marker, one EOC marker, and at least one tile-part (SOT and SOD). Each tile-part has one SOT and one SOD marker. The SOC, SOD, and EOC delimiting markers are 16 bits in length with no explicit length information.

A.4.1 Start of codestream (SOC)

Function: Marks the beginning of a codestream specified in this Recommendation | International Standard.
Usage: This is the first marker in the codestream. There shall be only one SOC per codestream.
Length: Fixed.

SOC: Marker value.

Table A-4 — Start of codestream parameter values

Parameter	Size (bits)	Values
SOC	16	0xFF4F

A.4.2 Start of tile-part (SOT)

Function: Marks the beginning of a tile-part and the index of its tile within a codestream. The tile-parts of a tile shall appear in order (see TPsot) in the codestream, but not necessarily consecutively.

Usage: Shall be the first marker segment in a tile-part header. There shall be at least one SOT in a codestream. There shall be only one SOT per tile-part.

Length: Fixed.

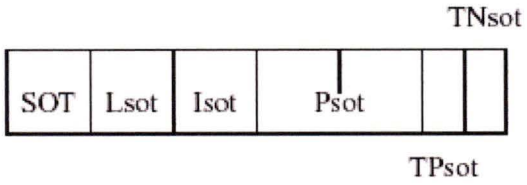


Figure A-5 — Start of tile-part syntax

SOT: Marker value. Table A-5 shows the size and values for start of tile-part.

Lsot: Length of marker segment in bytes (not including the marker).

Isot: Tile number. This number refers to the tiles in raster order starting at the number 0.

Psot: Length, in bytes, from the beginning of the first byte of this SOT marker segment of the tile-part to the end of the data of that tile-part. Figure A-13 shows this alignment. Only the last tile-part in the codestream may contain a 0 for Psot. If the Psot is 0, this tile-part is assumed to contain all data until the EOC marker.

TPsot:Tile-part instance. If this is a tile-part, there is a specific order required for decoding tile-parts; this index then denotes the order from 0. If there is only one tile-part for a tile then this value is zero. The tile-parts of this tile shall appear in the codestream in this order, although not necessarily consecutively.

TNsot:Number of tile-parts of a tile in the codestream. Two values are allowed: the correct number of tile-parts for that tile and zero. A zero value indicates that the number of tile-parts of this tile is not defined in this tile-part.

Table A-5 — Start of tile-part parameter values

Parameter	Size (bits)	Values
SOT	16	0xFF90
Lsot	16	10
Isot	16	0 — 65 535
Psot	32	0 — $(2^{32}-1)$
TPsot	8	0 — 255
TNsot	8	0 — 255

Table A-6 — Number of tile-parts, TNsot, parameter value

Value	Number of tile-parts
0	Number of tile-parts of this tile in the codestream is not defined

Table A-6 — Number of tile-parts, TNsot, parameter value

Value	Number of tile-parts
1 — 255	Number of tile-parts of this tile in the codestream

A.4.3 Start of data (SOD)

Function: Indicates the beginning of bit stream data for the current tile-part. The SOD also indicates the end of a tile-part header.

Usage: Shall be the last marker in a tile-part header. Data between an SOD and the next SOT or EOC (end of image) shall be a multiple of 8 bits — the codestream is padded with bits, as needed (see Annex D.4.2). There shall be at least one SOD in a codestream. There shall be one SOD per tile-part.

Length: Fixed.

SOD: Marker value

Table A-7 — Start of data parameter values

Parameter	Size (bits)	Values
SOD	16	0xFF93

A.4.4 End of codestream (EOC)

Function: Indicates the end of the codestream.

NOTE — This marker shares the same number as the EOI marker in ITU-T Rec. T.81 | ISO/IEC 10918-1.

Usage: Shall be the last marker in a codestream. There shall be one EOC per codestream.

Length: Fixed.

EOC: Marker value

Table A-8 — End of codestream parameter values

Parameter	Size (bits)	Values
EOC	16	0xFFD9

A.5 Fixed information marker segment

This marker segment describes required information about the image.The SIZ marker segment is required in the main header immediately after the SOC marker segment.

A.5.1 Image and tile size (SIZ)

Function: Provides information about the uncompressed image such as the width and height of the reference grid, the width and height of the tiles, the number of components, component bit depth, and the separation of component samples with respect to the reference grid.

Usage: There shall be one and only one in the main header immediately after the SOC marker segment. There shall be only one SIZ per codestream.

Length: Variable depending on the number of components.

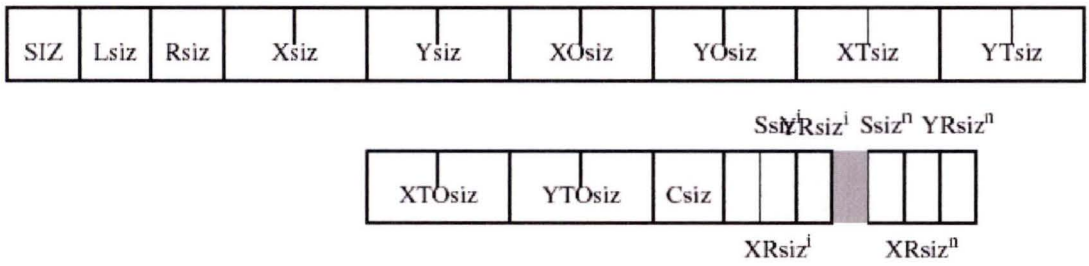


Figure A-6 — Image and tile size syntax

SIZ: Marker value. Table A-9 shows the size and parameter values for image and tile size.

Lsiz: Length of marker segment in bytes (not including the marker).

Rsiz: Denotes capabilities of the codestream.

Xsiz: Width of the reference grid.

Ysiz: Height of the reference grid.

XOsiz:Horizontal offset from the origin of the reference grid to the left side of the image area.

YOsiz:Vertical offset from the origin of the reference grid to the top side of the image area.

XTsiz:Width of one reference tile with respect to the reference grid.

YTsiz:Height of one reference tile with respect to the reference grid.

XTOsiz:Horizontal offset from the origin of the reference grid to the left side of the first tile.

YTOsiz:Vertical offset from the origin of the reference grid to the top side of the first tile.

Csiz: Number of components in the image.

Ssizi: Precision (depth) in bits and sign of the ith component. The precision is the precision of the component before the RCT or ICT is performed. (It is not necessarily the precision of the component plane coded in the file. The ICT or RCT can change the precision.) There is one occurrence of this parameter for each component. This parameter signals the component precision that is in the codestream. Only those bit-planes necessary need be extracted.
XRsizi:Horizontal separation of a sample of ith component with respect to the reference grid. There is one occurrence of this parameter for each component.

YRsizi:Vertical separation of a sample of ith component with respect to the reference grid. There is one occurrence of this parameter for each component.

Table A-9 — Image and tile size parameter values

Parameter	Size (bits)	Values
SIZ	16	0xFF51
Lsiz	16	42 — 49 191
Rsiz	16	use Table A-10
Xsiz	32	$1 — (2^{32} - 1)$
Ysiz	32	$1 — (2^{32} - 1)$
XOsiz	32	$0 — (2^{32} - 2)$
YOsiz	32	$0 — (2^{32} - 2)$
XTsiz	32	$1 — (2^{32} - 1)$
YTsiz	32	$1 — (2^{32} - 1)$
XTOsiz	32	$0 — (2^{32} - 2)$
YTOsiz	32	$0 — (2^{32} - 2)$
Csiz	16	1 — 16 384
$Ssiz^i$	8	use Table A-11
$XRsiz^i$	8	1 — 255
$YRsiz^i$	8	1 — 255

Table A-10 — Capability Rsiz parameter

Value (bits) MSB LSB	Capability
0000 0000 0000 0000	Capabilities specified in this Recommendation | International Standard only
	All other values reserved

Table A-11 — Component Ssiz parameter

Values (bits) MSB LSB	Coefficient size
x000 0000 x010 0101	Components are value+1; from 1 bit deep through 38 bits deep respectively (including the sign bit, if appropriate)[a]
0xxx xxxx	Components are unsigned values
1xxx xxxx	Components are signed values

Table A-11 — Component Ssiz parameter

Values (bits) MSB LSB	Coefficient size
	All other values reserved.

a. The component precision is limited by the number of guard bits, quantization, growth of coefficients at each level of the transform, and the number of coding passes that can be signalled. Not all combinations of coding styles will allow the coding of 38 bit samples.